Bachelor Thesis

Die Manipulationskomplexität des Wahlverfahrens von Schulze

The Complexity of Manipulating Schulze Voting

submitted to

Universität
Konstanz

Department of Computer & Information Science

by

Julian Müller

1st Referee: PD Dr. Sven Kosub
2nd Referee: Prof. Dr. Ulrik Brandes
February, 2013

Benutzungshinweis
☒ Benutzung möglich
ab 17.4.14
☐ Keine Benutzung möglich

Bachelorarbeit (3. Exemplar)
von Herrn Julian Müller
erhalten am 06.02.2013.2013
Gutachter dieser Arbeit:
PD. Dr. S. Kosub
Prof. Dr. U. Brandes

Zentrales Prüfungsamt
(S. Lehmann)

Zusammenfassung

Die Schulze-Methode ist ein Wahlverfahren, das eine Reihe von erwünschten Kriterien von Wahlverfahren erfüllt. Das Wahlverfahren wird heute eine weiter Verbreitung gefunden, gerade in Organisationen, die hauptsächlich im Internet ihren Ursprung haben oder sich mit diesem beschäftigen. In letzter Zeit wurde begonnen, zu erforschen, inwieweit und wie schwer dieses Wahlverfahren manipulierbar ist. In dieser Bachelorarbeit werden wir Algorithmen präsentieren, die eine konstruktive oder destruktive Manipulation der Schulze-Methode für eine Gruppe von Manipulatoren in Polynomialzeit finden. Außerdem werden wir bestimmen, dass konstruktive Manipulation im Allgemeinen P-vollständig und destruktive Manipulation NL-vollständig ist.

Abstract

The Schulze method is a voting rule that satisfies many desirable properties. It has found widespread use and has especially been adopted by many organizations that originate from or are centered around the Internet. The manipulability of this voting rule has become a recent topic of research. In this Bachelor thesis, we will present algorithms that can find a constructive or destructive manipulation of the Schulze method for a group of manipulators in polynomial time. Additionally, we will establish that constructive manipulation for coalitions is P-complete in general and that destructive manipulation is NL-complete.

Contents

1 Introduction **4**
1.1 The Gibbard-Satterthwaite Theorem 4
1.2 Preventing Manipulation 5
1.3 Related Work 5

2 Preliminaries **7**
2.1 Orders and Equivalence Relations 7
2.2 Voting Rules 8
2.3 Computational Manipulation Problems 10
2.4 Complexity 13

3 The Schulze Method **15**
3.1 Ballot Counting and Defeat Strength 15
3.2 Definition 17
3.3 Correctness 19

4 Constructive Unique Winner Manipulation **21**
4.1 Unweighted Voting 21
  4.1.1 A Simple Necessary Condition 22
  4.1.2 Manipulation Thresholds 25
  4.1.3 The Restricting Link Rule 26
  4.1.4 The Incoming Paths Rule 29
  4.1.5 The Beat Threshold 33
  4.1.6 Computing a Manipulation 40
  4.1.7 Complexity 44
4.2 Weighted Voting 52

5 Constructive Co-Winner Manipulation **54**
5.1 Unweighted Voting 54
  5.1.1 Weak Manipulation Thresholds 55
  5.1.2 The Strongly Restricting Link Rule 56
  5.1.3 The Match Threshold 57
  5.1.4 Computing a Manipulation 58
  5.1.5 Complexity 61

5.2 Weighted Voting 63

6 Destructive Manipulation 64
6.1 Unweighted Voting 64
6.1.1 Computing a Manipulation 64
6.1.2 Complexity 68
6.2 Weighted Voting 70

7 Conclusion 72

1 Introduction

People have different points of view; their opinions differ, and they agree and disagree. Often, people have to decide on one common opinion of the whole group; they have to settle on one alternative out of many. In order to decide on a collective opinion, the group needs a mechanism which derives a collective opinion out of the many individual opinions. Such mechanisms are *voting rules.*

As a consequence of Arrow's Impossibility Theorem [Arr50], there is no perfect voting rule. For this reason, many different voting rules are in use today. All these voting rules have different advantages and disadvantages. A voting rule which has become popular in recent years is the *Schulze method*; e.g., it was adopted by the Debian Project, the Pirate Parties of Sweden and Germany and the Wikimedia Foundation [Sch11].

People do not only hold different opinions; they also try to make sure that the group follows their opinion. For this goal, they may lie about their order of preference of the candidates, i.e., manipulate the election, if it changes the result of the election in their favour. This behaviour is called tactical voting.

1.1 The Gibbard-Satterthwaite Theorem

Are there voting rules where tactical voting is not possible? The famous Gibbard-Satterthwaite Theorem [Gib73; Sat75] gives an answer to this question.

Definition 1.1.

1. A voting rule is *onto* if for any alternative c there is some voting profile such that the voting rule will choose c as the winner.
2. A voting rule is *manipulable* if some voter can improve the election outcome, i.e., an alternative he likes better is picked, by changing his stated order of preference from his real order of preference.
3. A voting rule is *strategy-proof* if the voting rule is not manipulable.
4. A voting rule is *dictatorial* if there is one voter whose favorite alternative is always picked as the winner.

The Gibbard-Satterthwaite Theorem then states:

Theorem 1.2 (Gibbard 1973, Satterthwaite 1975). *If there are at least 3 candidates and the voting rule is strategy-proof and onto, then it is dictatorial.*

Voting rules that are dictatorial or not onto are faulty, because they don't comply to democratic values. Therefore, all reasonable voting rules are manipulable.

1.2 Preventing Manipulation

Because voting rules in general are manipulable, strategies to prevent manipulation are required. There are different ways to prevent manipulation:

1. Restrict the allowed order of preferences that voters can vote. One such example are single-peaked preferences [Bla48]. For these, strategy-proof voting rules exist [Mou80]. On the other hand, whilst voters could misrepresent their preferences before, they are now deliberately required to misrepresent them.

2. Make it computationally hard to find a manipulation for a voter or a group of voters. Then, a voter or a group of voters may be unable to manipulate the election because it is too involved and expensive to find a manipulation.

The second possibility is the one examined in this Bachelor thesis. Specifically, we will explore whether it is tractable to find a constructive or destructive manipulation for the Schulze method. In a constructive manipulation scenario, manipulators try to make one specific candidate the unique winner or one of the (potential) winners. In the destructive manipulation scenario, manipulators try to prevent a candidate from becoming a winner.

1.3 Related Work

Computational complexity as a means to prevent manipulation has gained interest in research. The complexity of some voting rules has been studied in the last few years.

In early results, it was found that the second-order Copeland rule and the Single Transferable Vote rule is hard to manipulate for an unbounded number of candidates, even if only one voter tries to do so [BTT89; BO91].

The difficulty of the weighted constructive unique winner manipulation problem has been researched for many prominent voting rules in the last few years. In this scenario, the manipulators are trying to make one candidate the only winner of the election. Furthermore, the voters have weights associated with them: A vote by a voter of weight k counts as if k voters had given the vote. For a number of voting rules it has been shown that this problem is hard, even if there is only a small number of candidates [CSL07; HH07].

Some research was also done on the complexity of the problem if the voters are unweighted. Especially interesting are results for the Ranked Pairs rule and the minmax rule, for Markus Schulze published experimental results that the Schulze method agrees in most scenarios with the minmax voting rule and the Ranked Pairs rule [Sch11]. For the Ranked Pairs rule, it was shown to be hard to manipulate for an unbounded number of candidates, even if there is only one voter trying to manipulate; for the minmax rule, it was shown that this is the case as soon as there are two manipulators [Xia+09].

Recently, the complexity of manipulation for the Schulze method has become a subject of research. In [PX12], Parkes and Xia considered different varieties of manipulations and their difficulty if they are applied to the Schulze method. As one of their results, they showed that unweighted constructive unique winner manipulation can be done in polynomial time at least if there is only one manipulator. Moreover, they also proved that a destructive manipulation can be found in polynomial time, no matter how many manipulators there are. Also recently, Menton and Singh considered how difficult it is to manipulate the Schulze method in [MS12]. They argued that in the case of constructive co-winner manipulation, all manipulators can vote identically. Most recently, it was shown that any manipulation problem of the Schulze method is fixed-parameter tractable with respect to the number of candidates in [HLM12].

2 Preliminaries

In Chapter 1, we mentioned voting rules and we briefly discussed the manipulation problems we are interested in. This chapter will lay the formal foundations for what we consider a voting rule, and we will formally define the manipulation problems. Furthermore, we will briefly discuss the complexity classes that are used in this thesis later on.

2.1 Orders and Equivalence Relations

To represent order of preferences, orders are needed. Foremost, we will use both strict and non-strict orders to represent these order of preferences, but we will also use them in other contexts. Furthermore, in some instances, equivalence relations, are used, too.

An order is a binary relation $R \subseteq A \times A$ on a domain A. A binary relation may have different properties:

Definition 2.1. Let $R \subseteq A \times A$ be a binary relation. Then,

1. R is *reflexive* $\iff (\forall a \in A)[(a,a) \in R]$.
2. R is *irreflexive* $\iff (\forall a \in A)[(a,a) \notin R]$.
3. R is *trichotomous* $\iff (\forall a,b \in A)[((a,b) \in R) \vee ((b,a) \in R) \vee (a = b)]$.
4. R is *transitive* $\iff (\forall a,b,c \in A)[(((a,b) \in R) \wedge ((b,c) \in R)) \rightarrow ((a,c) \in R)]$.
5. R is *negatively transitive* $\iff (A \times A) \setminus R$ is transitive .
6. R is *symmetric* $\iff (\forall a,b \in A)[((a,b) \in R) \rightarrow ((b,a) \in R)]$.
7. R is *antisymmetric* $\iff (\forall a,b \in A)[(((a,b) \in R) \wedge ((b,a) \in R)) \rightarrow (a = b)]$.

Using these properties of binary relations, orders and equivalence relations can be defined:

Definition 2.2. Let $R \subseteq A \times A$ be a binary relation.

1. R is a *total preorder* $\iff$ R is transitive, reflexive and trichotomous.
2. R is a *strict partial order* $\iff$ R is transitive and irreflexive.
3. R is a *strict weak order* $\iff$ R is a strict partial order and negatively transitive.
4. R is a *strict total order* $\iff$ R is a strict weak order and trichotomous.
5. R is an *equivalence relation* $\iff$ R is transitive, reflexive and symmetric.

For an easier understanding, examples are given for each type of order:

Example 2.1.

1. Let $R \subseteq \mathbb{N} \times \mathbb{N}$. Then, $(a, b) \in R \iff a \leq b \mod 2$ is a total preorder, $(a, b) \in R \iff a < b \mod 2$ is a strict weak order, and $(a, b) \in R \iff a = b \mod 2$ is an equivalence relation.
2. $<$ on the natural numbers $\mathbb{N}$ is a strict total order.
3. Let A be some finite set. Then, $\subset$ on $\mathcal{P}(A)$ is a strict partial order.

2.2 Voting Rules

In elections, there are candidates and voters who give their opinion of the candidates. Let C be the non-empty set of *candidates* (*alternatives*) and $n = \|C\|$ be the number of candidates.

Every voter gives a ranking of the candidates where he expresses which candidates he prefers to the others. In doing so, he may not order all candidates because he thinks two candidates are equivalent or he is unsure which one he prefers more. This is a *partial ranking*: A partial ranking is a strict partial order on C. He may also choose to order all candidates in some way. This is a *complete ranking*: A complete ranking is a strict total order on C. Note that a complete ranking is also a partial ranking. The family of all partial rankings will be denoted by $\mathcal{R}$, whilst the family of all complete rankings will be denoted by $\mathcal{T}$.

Normally, there isn't only one voter, but there are many voters participating in an election. The votes given by a group of voters form a voting profile. An *unweighted*[1] *voting profile* is a list of rankings given by some voters. In other words, it is a vector $V = (v_1, ..., v_m)$ where $v_1, ..., v_m \in \mathcal{R}$ are votes.

---

[1] If the type of voting profile is not explicitly stated later on, it is always an unweighted voting profile.

The voters may have been split in multiple groups originally. If we view these groups as one group of voters again, then their voting profiles will be combined into one. This is voting profile concatenation. The concatenation of two voting profiles $V = (v_1, ..., v_m)$ and $V = (v'_1, ..., v'_\ell)$ is the profile $V \circ V' = (v_1, ..., v_m, v'_1, ..., v'_\ell)$.

A *voting rule* is a method that determines the winner(s) of an election. The result of an election is determined by the votes of the voters. Moreover, there must be at least one winner. Hence, a voting profile can be seen as a (computable) function $r : \mathcal{R}^+ \to \mathcal{P}(C) \setminus \{\emptyset\}$ that takes the voting profile of all voters as input and produces the winners of the election as output.[2]

Not all voters may be equal. Sometimes, voters hold shares of the total number of voters (e.g., shareholders of a corporation), in other words their vote may count as more than one vote. The vote then not only contains the vote of the voter, but also information of how much this vote is worth. The worth of a vote is determined by its *voting weight.* For a group of voters, the voting weights are represented as a vector $W = (w_1, ..., w_m) \in \mathbb{N}_+^m$.

As we have seen, a weighted vote consists of a vote and its associated voting weight. Therefore, the weighted votes of a group of voters forms a weighted voting profile. A *weighted voting profile* is a pair (V, W) where $V = (v_1, ..., v_m)$ is an unweighted voting profile associated with voting weights $W = (w_1, ..., w_m)$. If there are two groups of voters, then we may have to combine their voting profiles. The concatenation of two weighted voting profiles (V, W) and (V', W') with $V = (v_1, ..., v_m)$, $V' = (v'_1, ..., v'_\ell)$, $W = (w_1, ..., w_m)$ and $W' = (w'_1, ..., w'_\ell)$ is

$$(V, W) \circ (V', W') = ((v_1, ..., v_m, v'_1, ..., v'_\ell), (w_1, ..., w_m, w'_1, ..., w'_\ell)) \ .$$

The voting weight tells us how many votes a single vote is worth, i.e., how often a single vote is counted. We can therefore produce an unweighted voting profile that corresponds to the weighted voting profile by duplicating a vote as often as its voting weight demands: A vote of weight k is treated as k votes. This unweighted profile is the *expansion* of the weighted profile. Formally, the expansion $E(V, W) \in \mathcal{R}^{\sum_{i=1}^m w_i}$ of a weighted m-voter voting profile $(V, W) = ((v_1, ..., v_m), (w_1, ..., w_m))$ is the unweighted profile that satisfies

$$E(V, W)_i := v_{\max\{j \in \{1,...,m\} \mid \sum_{\ell=1}^{j-1} w_\ell < i\}} \text{ for all } i \in \{1, ..., \sum_{\ell=1}^{m} w_\ell\} \ . \tag{2.1}$$

If a voting rule is applied to a weighted voting profile (V, W), then it is applied to its expansion $E(V, W)$.

---

[2] As done in many similar publications, we will disregard Tie-Breaking, because it can fundamentally change the computational hardness of the manipulation (cf. [OE11] and [OEH11]). For this reason, we will not differentiate between voting rules, which only produce one candidate as a result, and voting correspondences, which produce a set of candidates as a result, but will only use the latter concept under the name voting rule.

Sometimes, we have to determine the expansion of a weighted voting profile that is the concatenation of two other weighted voting profiles (V, W) and (V', W'). Then, it doesn't matter whether we first concatenate the weighted profiles and that determine the expansion of the conatenated profile, or whether we first determine the expansions of (V, W) and (V', W') and then concatenate both expansions: Both result in the same profile. Formally, this means that $E(V, W) \circ E(V', W') = E((V, W) \circ (V', W'))$.

2.3 Computational Manipulation Problems

Voters have different goals in the election and differ in how they want to influence the election result. Accordingly, the type of manipulation they try to implement differs with their goals.

In addition, the situation differs for manipulating voters in what they know of the behaviour of the other voters and in what they are permitted to do to enforce their manipulation.

In our scenarios, the manipulation of the manipulating voters is subject to the following constraints:

- There is a group of manipulators and a group of non-manipulators which form the electorate. All manipulators share the same goal in their manipulation effort.
- The manipulators know how many non-manipulators there are and how each non-manipulator has voted. To phrase it differently, they have complete information on the behaviour of the non-manipulators. There is no secret ballot.
- The manipulators form a coalition. They can agree collectively what ballot each manipulator will vote in the election.
- The manipulators cannot introduce fake ballots. They cannot change the non-manipulators' votes.
- The manipulators' goal concerns the position of one special candidate. They aren't interested in the results of the other candidates, as long as they do not prevent the manipulators' goal concerning the special candidate.

According to these constraints, the manipulators' decision how to manipulate is impacted by the non-manipulators' voting profile, by the manipulators themselves (and, in the weighted case, each manipulator's voting weight), and the special candidate whose position the manipulators want to change in their goal's favour.

There is no other factor that influences the manipulators' votes. Therefore, we will call the non-manipulators voting profile, the manipulator coalition and the special candidate a manipulation instance.

Definition 2.3.

1. An *unweighted manipulation instance* is a triple (V_{N}, M, p) where V_{N} is the non-manipulators' unweighted voting profile, M is the finite, non-empty set of manipulators, and $p \in C$ is the special candidate whose position in the election result the manipulators wish to influence.

2. A *weighted manipulation instance* is a quintuple $(V_{\mathrm{N}}, W_{\mathrm{N}}, M, W_{\mathrm{M}}, p)$ where $(V_{\mathrm{N}}, W_{\mathrm{N}})$ is the non-manipulators' weighted voting profile, M is the finite, non-empty set of manipulators, $W_{\mathrm{M}} \in \mathbb{N}_{+}^{k}$ are the manipulators' voting weights, and $p \in C$ is the candidate who the manipulators are concerned about.

Besides the manipulation instance, the manipulators' goal decides how the manipulators will vote. As described above, the manipulators' goal only concerns one special candidate in our scenarios. There are two possibilities what the manipulators want to do: They could try to improve the candidate's election result, or they could try to deteriorate the candidate's result. The first possibility implies that they attempt a *constructive manipulation*, the latter means that they attempt a *destructive manipulation*.

However, we will not look at whether the manipulators can improve or deteriorate a candidate's election result at all, but whether the manipulators can enforce a specific result for the candidate.

- The manipulators' goal is to make the candidate a winner. This is *constructive co-winner manipulation*.

- As a stronger version of the previous goal, the manipulators try to make the candidate the only winner of the election, i.e., there is no other winner besides the manipulators' preferred candidate. This type of manipulation is called *constructive unique winner manipulation*.

- The manipulators' goal is to prevent the candidate from winning. We will call this *destructive manipulation*.

Now, we can define what these different types of manipulations mean in a mathematical sense:

Definition 2.4. Let r be a voting rule. Let $I = (V_{\mathrm{N}}, M, p)$ be an unweighted manipulation instance with $k = \|M\|$ manipulators. Let $V_{\mathrm{M}} \in \mathcal{R}^k$ be an unweighted voting profile for the manipulators.

A pair (I, V_{M}) is

1. an *unweighted constructive unique winner manipulation* if and only if p is the unique winner of the election for these votes, i.e., $r(V_{\mathrm{N}} \circ V_{\mathrm{M}}) = \{p\}$.
2. an *unweighted constructive co-winner manipulation* if and only if p is a winner of the election for these votes, i.e., $r(V_{\mathrm{N}} \circ V_{\mathrm{M}}) \ni p$.
3. an *unweighted destructive manipulation* if and only if p is no winner of the election for these votes, i.e., $r(V_{\mathrm{N}} \circ V_{\mathrm{M}}) \not\ni p$.

Definition 2.5. Let r be a voting rule. Let $I = (V_{\mathrm{N}}, W_{\mathrm{N}}, M, W_{\mathrm{M}}, p)$ be a weighted manipulation instance. Let $(V_{\mathrm{M}}, W_{\mathrm{M}})$ be a weighted voting profile for the manipulators.

A pair (I, V_{M}) is

1. a *weighted constructive unique winner manipulation* if and only if p is the unique winner of the election for these votes, i.e., $r((V_{\mathrm{N}}, W_{\mathrm{N}}) \circ (V_{\mathrm{M}}, W_{\mathrm{M}})) = \{p\}$.
2. a *weighted constructive co-winner manipulation* if and only if p is a winner of the election for these votes, i.e., $r((V_{\mathrm{N}}, W_{\mathrm{N}}) \circ (V_{\mathrm{M}}, W_{\mathrm{M}})) \ni p$.
3. a *weighted destructive manipulation* if and only if p is no winner of the election for these votes, i.e., $r((V_{\mathrm{N}}, W_{\mathrm{N}}) \circ (V_{\mathrm{M}}, W_{\mathrm{M}})) \not\ni p$.

For each type of manipulation, we can then define the corresponding computational decision problems:

Definition 2.6 (UNWEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION). Let r be a voting rule, and I be an unweighted manipulation instance. Is there a corresponding unweighted constructive unique winner manipulation?

Definition 2.7 (WEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION). Let r be a voting rule, and I be an weighted manipulation instance. Is there a corresponding weighted constructive unique winner manipulation?

The other constructive goal may be that the preferred candidate should at least become one of the (potentially many) winners. The corresponding computational problems are defined below:

Definition 2.8 (UNWEIGHTED CONSTRUCTIVE CO-WINNER MANIPULATION). Let r be a voting rule, and I be an unweighted manipulation instance. Is there a corresponding unweighted constructive co-winner manipulation?

Definition 2.9 (WEIGHTED CONSTRUCTIVE CO-WINNER MANIPULATION). Let r be a voting rule, and I be an weighted manipulation instance. Is there a corresponding weighted constructive co-winner manipulation?

Definition 2.10 (UNWEIGHTED DESTRUCTIVE MANIPULATION). Let r be a voting rule, and I be an unweighted manipulation instance. Is there a corresponding unweighted destructive manipulation?

Definition 2.11 (WEIGHTED DESTRUCTIVE MANIPULATION). Let r be a voting rule, and I be an weighted manipulation instance. Is there a corresponding weighted destructive manipulation?

The voting profiles of the manipulators or the non-manipulators in the manipulation problem can also be restricted to complete rankings.

2.4 Complexity

As part of this work, we will study how hard or how easy the different manipulation problems are. For this, we will determine the complexity classes of the different manipulation problems. We will only consider the decision problem version; in other words, we will only look how difficult it is to test if there is a successful manipulation.

This section only provides a short overview of the complexity classes and related notions that we need to determine the complexity of constructive and destructive manipulation. For a more elaborated discussion, see [Pap94].

In general, decision problems which can be solved in polynomial running time on a deterministic Turing machine are normally considered efficiently solvable. All these problems belong to the complexity class P. There are also problems whose solutions can be verified in polynomial time on a deterministic Turing machine; all these problems form to the complexity class NP. It is a long-standing open question if all problems in NP are efficiently solvable, i.e., if P = NP.

Additionally, we will look at the complexity class NL. The class NL contains all decision problems which can be decided by a non-deterministic Turing machine in logarithmic space. In the same manner, the class L contains all decision problems which can be solved by a deterministic Turing machine in logarithmic space.

Note that all decision problems in both of these classes are especially inherently parallelizable.[3]

Of special interest are the hardest problems in the complexity classes. A problem is NP-hard if all other problems in NP can be reduced to using a polynomial time reduction. For the complexity class P and its sub-classes we need a weaker form of reduction. We will call a problem P-hard or NL-hard if there is a logarithmic space reduction from all other problems in this class to this problem. A problem is P-complete, if it is in P and P-hard. Similarly, we can also define NP-completeness and NL-completeness.

P-complete problems are HORNSAT (cf. [Pap94, p.176, Corollary]) and its variant DUAL HORNSAT. HORNSAT and DUAL HORNSAT have simple reductions to each other: If we negate all variables of a Horn formula, we get a dual Horn formula, and vice versa.

Definition 2.12 (DUAL HORNSAT)**.** We are given a formula F consisting of dual Horn clauses $C_1, ..., C_m$. A dual Horn clause is a clause where at most one literal is negated. Is F satisfiable?

The canonical NL-complete problem is the *st*-connectivity problem STCON (cf. [Pap94, p.398, Theorem 16.2]):

Definition 2.13 (STCON)**.** We are given a directed graph $G = (V, E)$ and two vertices $s, t \in V$. Is there a path in G from s to t?

---

[3] In particular, there is a CRCW PRAM with a polynomial number of processors that solves these problems in $\mathcal{O}(\log n)$ time.

3 The Schulze Method

The Schulze method is a voting rule that was first proposed by Markus Schulze in 1997. This chapter will examine how the Schulze method is defined, and it will prove that the Schulze method is a voting rule.

The Schulze method is only defined here for the unweighted case. If the votes are weighted, then the Schulze method is applied to the expansion of the weighted voting profile.

This chapter is based on [Sch11].

3.1 Ballot Counting and Defeat Strength

Just like many voting rules, the Schulze method counts ballots. In the case of the Schulze method, the Schulze method uses pairwise counting, i.e., it is counted in how many ballots one candidate is strictly preferred to the other. The result of the ballot counting is the *Condorcet pairwise matrix*. The Condorcet pairwise matrix will be denoted by the letter $N = N(V) \in \mathbb{N}^{C \times C}$ where V is a voting profile. If the voting profile V consists of the votes $V = (v_1, ..., v_m)$, then N_{ab} contains the number of voters which strictly prefer candidate a to candidate b, i.e., the Condorcet pairwise matrix has the form

$$N_{ab} = \|\{i \in \{1, ..., m\} | a >_{v_i} b\}\| \qquad \text{for all } a, b \in C \; . \tag{3.1}$$

A pair (a, b) of candidates is a *link*. N_{ab} is the *support* of the link, whereas N_{ba} is its *opposition*.

The Schulze method uses the strength of links to decide which candidate wins. Therefore, some metric is needed to determine how strong a link is. This metric is a *measure of defeat strength* which assigns a strength to a link, so it is a function $d : C \times C \to \mathbb{R}$. One measure of defeat strength is the *margins* measure

$$d_{\mathrm{m}}(a, b) = N_{ab} - N_{ba} \; . \tag{3.2}$$

The strength of a link should not depend on the concrete identities of the compared candidates. Instead, the strength should only depend on the size of the support and the opposition of a link.

A link (a, b) is stronger than a link (b, a) if and only if $d(a, b) > d(b, a)$. The concrete values assigned as strength to a link do not matter; only the order of the links, i.e., which link is stronger, is important. As a measure of defeat strength should only depend on the support and the opposition of the link, we can also express it a as an ordering $>_D$ on the pairs (N_{ab}, N_{ba}) for all $a, b \in C$ where $(N_{ab}, N_{ba}) >_D (N_{cd}, N_{dc})$ if and only if $d(a,b) > d(c,d)$. This *defeat strength order* $>_D$ is a strict weak order on the domain $\mathbb{N} \times \mathbb{N}$.

A defeat strength order $>_D$ can have different properties. Here, we will especially need one common property of defeat strength: Monotonicity. $>_D$ is *(strictly) monotonic* if and only if an increase in the support, but a non-increase in opposition, or a decrease in opposition, but a non-decrease in support increases the strength of a link, i.e., if and only if for all $x_1, x_2, y_1, y_2 \in \mathbb{N}$

$$((x_1 > y_1 \land x_2 \leq y_2) \lor (x_1 \geq y_1 \land x_2 < y_2)) \implies (x_1, x_2) >_D (y_1, y_2) \; . \qquad (3.3)$$

It will be explicitly stated if we need that the defeat strength is monotonic.

Since the defeat strength order $>_D$ is a strict weak order, $a \sim_D b \iff a \not>_D b \land b \not>_D a$ forms an equivalence relation and $a \gtrsim_D b \iff a >_D b \lor a \sim_D b$ forms a total preorder.

The function $\min_D : \mathcal{P}_{\text{fin}}(\mathbb{N} \times \mathbb{N}) \setminus \{\emptyset\} \to \mathbb{N} \times \mathbb{N}$ determines an arbitrary, but fixed minimal element of a non-empty, finite set $A \subseteq \mathbb{N} \times \mathbb{N}$ according to the defeat strength order $>_D$, if applied to the set A. Correspondingly, the function $\max_D$ determines an arbitrary, but fixed maximal element of a non-empty, finite set $A \subseteq \mathbb{N} \times \mathbb{N}$.

If V is a voting profile, then $D = D(V) \in (\mathbb{N} \times \mathbb{N})^{C \times C}$ is the *defeat strength pairwise matrix* with

$$D_{ab} = (N_{ab}, N_{ba}) \qquad \text{for all } a, b \in C \; . \qquad (3.4)$$

Correspondingly, $M = M(V) \in \mathbb{Z}^{C \times C}$ is the *margins pairwise matrix* for the margins measure, with

$$M_{ab} = d_{\text{m}}(a,b) = N_{ab} - N_{ba} \; . \qquad (3.5)$$

We will use the margins measure especially in concrete examples. In these examples, we depict the voting profiles, or rather the margins pairwise matrix, by a *pairwise comparison graph.* The pairwise comparison graph consists of all candidates and has a directed edge (a, b) from candidate a to candidate b if and only if a had a pairwise victory against b, i.e., the support of the link (a, b) is greater than its opposition. We associate with such a directed edge the defeat strength as measured by the margins measure.[1] Thus, we can unambiguously determine the margins pairwise matrix from a pairwise comparison graph.

---

[1] The edge weight could also be determined by any other measure of defeat strength. However, other measures are never used in this thesis when a pairwise comparison graph is shown.

If two voting profiles V and V' are combined, then the Condorcet pairwise matrices add, i.e., $N(V \circ V') = N(V) + N(V')$. For this reason, the defeat strength pairwise matrices and the margins pairwise matrices add, too: $D(V \circ V') = D(V) + D(V')$, using vector addition in each entry, and $M(V \circ V') = M(V) + M(V')$.

Pseudo-votes are a helpful tool. Even though a voter cannot give a ranking that translates into the pseudo-vote, they are quite useful in the analysis of the Schulze method. The defeat strength matrix I^+ stands for a voter who supported all links (even opposing ones), while the matrix I^- stands for a voter who opposed all links. Formally, these matrices can be defined as

$$I^+ = (i^+_{ij})_{i,j \in C} \text{ with } i^+_{ij} = (1,0) \text{ and} \tag{3.6}$$

$$I^- = (i^-_{ij})_{i,j \in C} \text{ with } i^-_{ij} = (0,1) \ . \tag{3.7}$$

Accordingly, kI^+ and kI^- stand for k pseudo-votes as defined above.

For the margins measure, the equivalent of the I^+ matrix is the matrix of ones, and the equivalent of the I^- matrix is the negative of the matrix of ones.

3.2 Definition

The Schulze method will be defined using the defeat strength pairwise matrix D with the defeat strength order $>_D$. It can also be adapted in a straightforward way to the margins pairwise matrix.

Definition 3.1.

1. An (a,b)-*path* $P = (c_0, ..., c_\ell)$ is a sequence of candidates $c_0, ..., c_\ell \in C$ such that $c_0 = a$ and $c_\ell = b$.
2. The *strength of a path* $P = (c_0, ..., c_\ell)$ is the strength of its weakest link, i.e.,

$$s_D(P) = \min_D\{D_{c_{i-1}c_i} | i \in \{1, ..., \ell\}\} \ . \tag{3.8}$$

3. The *path strength function* $S_D : C \times C \to \mathbb{N} \times \mathbb{N}$ contains the strength of the strongest (a,b)-path, i.e.,

$$S_D(a,b) = \max_D\{s_D(P') | P' \text{ is } (a,b)\text{-path}\} \ . \tag{3.9}$$

We can use the path strength function to define a relation on the candidates and the set of winners:

Definition 3.2. A candidate a is ranked above a candidate b in the *Schulze ranking*, $a >_{\text{Schulze}} b$, if and only if

$$S_D(a,b) >_D S_D(b,a) \ . \tag{3.10}$$

The *set of winners* is the set of candidates $c \in C$ above whom no other candidate is ranked in the Schulze ranking, i.e., there is no candidate $b \in C$ with $b >_{\text{Schulze}} c$.

The following example shows the winners of an election according to the Schulze method, using margins as the measure of defeat strength:

Example 3.1. In the election, there are 4 candidates $C = \{a,b,c,d\}$. The voters gave ballots as seen in Table 3.1 First, we need to count the ballots and calculate the defeat strength from the counts. The resulting strengths of the links are shown in Fig. 3.1.

Table 3.1: Votes by voters in Example 3.1.

Number voters	Ranking
4	$a > b > d > c$
6	$a > d > c > b$
5	$b > a > d > c$
2	$b > d > a > c$
4	$c > d > b > a$
7	$c > b > a > d$
3	$d > a > c > b$

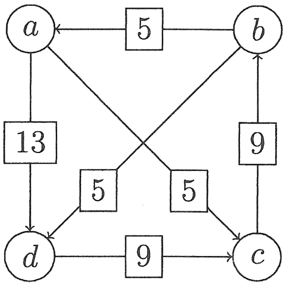

Figure 3.1: Pairwise comparison graph of Example 3.1.

Any path to a has at most strength 5 because the link (b,a) has strength 5. Additionally, the (a,d)-path (a,d) has strength 13, the (a,c)-path (a,d,c) has strength 9, and finally the (a,b)-path (a,d,c,b) has strength 9. Thus, a is never defeated, but beats all other candidates, so a is the unique winner of this election.

3.3 Correctness

Although the set of winners was defined in the previous section, it is not yet known if there is always at least one winner. If there were no winner according to the Schulze method's definition, the Schulze ranking would be either not irreflexive or not transitive, since the number of candidates is finite. As it is clearly irreflexive by definition, it only remains to be shown that the relation is transitive.

Theorem 3.3. *The Schulze ranking is transitive.*

Proof. Let $a, b, c \in C$ be candidates. The Schulze ranking is transitive if the following holds:

$$((S_D(a,b) >_D S_D(b,a)) \wedge (S_D(b,c) >_D S_D(c,b))) \implies S_D(a,c) >_D S_D(c,a) \; . \tag{3.11}$$

So suppose that

$$S_D(a,b) >_D S_D(b,a) \; , \tag{3.12}$$
$$S_D(b,c) >_D S_D(c,b) \; . \tag{3.13}$$

Because the strongest (x, y)-path for some candidates $x, y \in C$ is at least as strong as the strongest (x, y)-path containing some candidate $z \in C$, we can infer the following inequations:

$$\min_D\{S_D(c,a), S_D(a,b)\} \lesssim_D S_D(c,b) \; , \tag{3.14}$$
$$\min_D\{S_D(a,b), S_D(b,c)\} \lesssim_D S_D(a,c) \; , \tag{3.15}$$
$$\min_D\{S_D(b,c), S_D(c,a)\} \lesssim_D S_D(b,a) \; . \tag{3.16}$$

Consider the following two cases:

1. Suppose it holds that
$$S_D(a,b) \gtrsim_D S_M(b,c) \; . \tag{3.17}$$
Equations (3.12) and (3.17) lead to
$$S_D(a,b) >_D S_D(c,b) \; . \tag{3.18}$$
From (3.14) and (3.18) we obtain that
$$S_D(c,a) \lesssim_D S_D(c,b) \; . \tag{3.19}$$
From (3.15) and (3.17) follows
$$S_D(b,c) \lesssim_D S_D(a,c) \; . \tag{3.20}$$
Combining (3.19), (3.13) and (3.20) gives:
$$S_D(c,a) \lesssim_D S_D(c,b) <_D S_D(b,c) \lesssim_D S_D(a,c) \; . \tag{3.21}$$

2. Suppose it holds that
$$S_D(a,b) <_D S_D(b,c) \ . \tag{3.22}$$
Equations (3.13) and (3.22) lead to
$$S_D(b,c) >_D S_D(b,a) \ . \tag{3.23}$$
From (3.16) and (3.23) follows that
$$S_D(c,a) \lesssim_D S_D(b,a) \ . \tag{3.24}$$
From (3.15) and (3.22) we obtain that
$$S_D(a,b) \lesssim_D S_D(a,c) \ . \tag{3.25}$$
Combining (3.24), (3.12) and (3.25) gives:
$$S_D(c,a) \lesssim_D S_D(b,a) <_D S_D(a,b) \lesssim_D S_D(a,c) \ . \tag{3.26}$$

Altogether, $S_D(a,c) >_D S_D(c,a)$, so $>_{\text{Schulze}}$ is transitive. □

In other words, the better relation is a strict partial order of the candidates. Based on this thought, the winners can also be characterized as the maximal elements of the strict partially ordered set $(C, >_{\text{Schulze}})$. There is also another way to characterize winners:

Proposition 3.4.

1. *A candidate $c \in C$ is a winner if and only if $S_D(c,b) \gtrsim_D S_D(b,c)$ for all $b \in C \setminus \{c\}$.*
2. *A candidate $c \in C$ is a unique winner if and only if $S_D(c,b) >_D S_D(b,c)$ for all $b \in C \setminus \{c\}$.*

Proof.

1. Suppose $c \in C$ is not a winner. Then, there is a candidate $b \in C \setminus \{c\}$ with $b >_{\text{Schulze}} c$. Hence, $S_D(b,c) >_D S_D(c,b)$.

   Suppose $c \in C$ is a winner. Then, there is no candidate $b \in C \setminus \{c\}$ with $b >_{\text{Schulze}} c$. Therefore, $S_D(b,c) \lesssim_D S_D(c,b)$ for all candidates $b \in C \setminus \{c\}$.

2. Suppose $c \in C$ is a unique winner and there is a candidate $b \in C \setminus \{c\}$ with $S_D(b,c) \sim_D S_D(c,b)$. If b is a maximal element of $(C, >_{\text{Schulze}})$, then there are two winners, which is a contradiction. Otherwise, there is a maximal element a of $(C, >_{\text{Schulze}})$ with $a >_{\text{Schulze}} b$, i.e., $S_D(b,a) >_D S_D(a,b)$. Since $S_D(b,c) \sim_D S_D(c,b)$, $a \neq c$, and there are two winners, which is a contradiction.

   Suppose there is a candidate $c \in C$ such that $S_D(c,b) >_D S_D(b,c)$ for all candidates $b \in C \setminus \{c\}$. Then, $c >_{\text{Schulze}} b$ for all candidates $b \in C \setminus \{c\}$. Hence, b is the only maximal element of $(C, >_{\text{Schulze}})$ and the unique winner. □

4 Constructive Unique Winner Manipulation

In the first chapter of the series of chapters about specific manipulation problems, we start out with a constructive manipulation problem: The manipulators aim to make one, specifically chosen, candidate the one and only winner of the election. Our task is to decide whether there is a successful manipulation, and, if there is one, to calculate one so that the manipulators are able to give their votes accordingly.

This manipulation problem comes in two flavours: Either all voters have the same voting weight, i.e., we have a scenario of unweighted voting, or the voters have different voting weights, i.e., we have an instance of weighted voting. We will begin with the unweighted voting problem, for it is a special case of the weighted voting problem.

4.1 Unweighted Voting

In Sect. 2.3, we defined in Definition 2.4 what a solution to the unweighted constructive unique winner manipulation problem is. However, the formulation was a general one for any voting rule. It can also be expressed directly for the Schulze method:

Proposition 4.1. *Let* $I = (V_\mathrm{N}, M, p)$ *be an unweighted manipulation instance with* $k = \|M\|$ *manipulators.*

A pair (I, V_M) *is an unweighted constructive unique winner manipulation if and only if*

$$S_{D(V_\mathrm{N} \circ V_\mathrm{M})}(p, c) >_D S_{D(V_\mathrm{N} \circ V_\mathrm{M})}(c, p) \text{ for all } c \in C \setminus \{p\} \; . \tag{4.1}$$

Proof. This is a direct consequence of Proposition 3.4. ☐

This characterization of an unweighted constructive unique winner manipulation for the Schulze method, however, only allows us to check if a specific voting profile is a manipulation, but we are unable to decide if a successful manipulation exists and, if one exists, to find it in polynomial time only using Proposition 4.1.

Henceforth, we will try to find a criterion that can tell us if there exists an unweighted constructive unique winner manipulation for the specific manipulation instance. It is difficult to find such a criterion, though, if we allow all possible defeat strength measures. Therefore, we will only look at monotonic defeat strength measures.

4.1.1 A Simple Necessary Condition

In the following, we assume that there are k manipulators that try to manipulate the voting results, that they try to make p win, and that the non-manipulators have voted a voting profile $V = V_{\mathrm{N}}$ that has the associated pairwise defeat strength matrix D.

If the measure of defeat strength is monotonic, we are able to easily determine the minimal and maximal strength of a (simple) path over all possible manipulators' voting profiles: The strength of a simple path is minimal for all profiles if all manipulators objected to all traversed links, i.e., we can determine the minimum path strength for all candidates $a, b \in C$ over all manipulators' voting profiles $V_{\mathrm{M}} \in \mathcal{R}^k$ by

$$S_{D+kI^-}(a,b) \sim_D \min{}_D\{S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(a,b) | V_{\mathrm{M}} \in \mathcal{R}^k\} \; . \tag{4.2}$$

Furthermore, for a monotonic measure of defeat strength, the strength is maximal for all profiles if all manipulators supported all traversed links. Therefore, we can also determine the maximum path strength over all manipulators' voting profiles by

$$S_{D+kI^+}(a,b) \sim_D \max{}_D\{S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(a,b) | V_{\mathrm{M}} \in \mathcal{R}^k\} \; . \tag{4.3}$$

We can base a necessary condition on this observation: If there is an unweighted constructive unique winner manipulation, there must be a manipulators' voting profile such that the strongest path from candidate p to some other candidate c is stronger than the strongest path from candidate c to candidate p. The strongest (p, a)-path for the manipulating profile is at most as strong as the strongest (p, a)-path over all possible voting profiles, and the strongest (a, p)-path is at least as strong as the strongest (p, a)-path for some voting profile such that the strength of the strongest (p, a)-path is minimal. Our observation for monotonic measures of defeat strength, however, gives us an easy method to calculate the strongest path over all voting profiles and the strongest path with minimal strength over all voting profiles. This yields the following necessary condition.

Proposition 4.2. *Let (V, M, p) be an unweighted manipulation instance with $k = \|M\|$ manipulators. Let $D = D(V)$. Let the defeat strength be monotonic.*

If there is an unweighted constructive unique winner manipulation, then

$$S_{D+kI^+}(p,c) >_D S_{D+kI^-}(c,p) \text{ for all } c \in C \setminus \{p\} \; . \tag{4.4}$$

Remark 4.1. In particular, the necessary condition takes the following form if the measure of defeat strength is margins:

$$S_M(p,c) + k > S_M(c,p) - k \text{ for all } c \in C \setminus \{p\} \ . \tag{4.5}$$

Example 4.2. We measure the defeat strength using the margins method. There is only one manipulator, and the votes of the non-manipulators resulted in the following pairwise comparison graph: The strength of a path from p to some other candidate can become stronger by at most 1 in comparison to its strength in Fig. 4.1, for there is only one manipulator. Therefore, the maximum strength from p to b is $S_M(p,b) + 1 = 7$. The path from b to p can be reduced by at most one, so we also know that the path strength from b to p is at least $S_M(b,p) - 1 = 9$. Hence, there is no unweighted constructive unique winner manipulation, for the strongest (p,b)-path is always weaker than the strongest (b,p)-path.

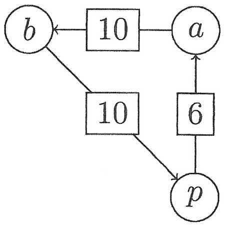

Figure 4.1: Pairwise comparison graph for Example 4.2.

We can use the necessary condition in Proposition 4.2 at least to decide for some manipulation instances that there is no unweighted constructive unique winner manipulation. However, we tried to find a criterion in order to decide for all manipulation instances whether there is an unweighted constructive unique winner manipulation. Does Proposition 4.2 deliver such a criterion? Parkes and Xia proved in [PX12] that it is a criterion at least for some manipulation instances:

Theorem 4.3 (Parkes, Xia 2012). *Let (V, M, p) be an unweighted manipulation instance such that there is exactly one manipulator, i.e., $k = \|M\| = 1$, and all non-manipulators vote complete rankings, i.e., $V \in \mathcal{T}^+$. Let $D = D(V)$. Let the defeat strength be monotonic.*

There is an unweighted constructive unique winner manipulation if and only if

$$S_{D+kI^+}(p,c) >_D S_{D+kI^-}(c,p) \textit{ for all } c \in C \setminus \{p\} \ . \tag{4.6}$$

However, Proposition 4.2 cannot be used if there is more than one manipulator or if the non-manipulators also vote non-complete rankings. In these cases, there are instances where this criterion fails as a sufficient condition:

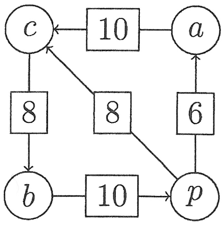

Figure 4.2: Pairwise comparison graph of Example 4.3.

Example 4.3. We use margins as our measure of defeat strength. There are two manipulators, and the votes of the non-manipulators resulted in the pairwise comparison graph in Fig. 4.2.

First, the criterion in Proposition 4.2 is tested. For that, we need to compute the path strength from p to any other candidate, if all manipulators support all links of the connecting paths, and the path strength from some candidate to p, if all manipulators oppose the links of connecting paths. For the margins measure, we can simply compute the path strengths from and to p after the non-manipulators have voted, and afterwards add or subtract the number of manipulators.

Table 4.1 shows the path strengths after the non-manipulators have voted, but before the manipulators gave their vote.

Table 4.1: Path strengths from and to p in Example 4.3.

x	a	b	c
$S_M(p, x)$	6	8	8
$S_M(x, p)$	8	10	8

As we test the necessary condition, we come to the conclusion that the necessary condition in Proposition 4.2 is satisfied: The maximum path strength from p to a is 8 after the manipulators' vote, whilst the minimum path strength from a to p is 6; the maximum path strength from p to b is 10, whilst the minimum path strength from b to p is 8; the maximum path strength from p to c is 10, whilst the minimum path strength from b to p is 6.

However, there cannot be an unweighted constructive unique winner manipulation in this example: Suppose there was one; then, p must beat b. Since the path strength from b to p is at least 8, the strongest (p, b)-path must have at least strength 9. The only fitting (p, b)-path is the path from p to b via c. Therefore, we also know that there is a (c, p)-path of at least strength 8. Consequently, there is also an (a, p)-path of at least strength 8, for the link (a, c) has also at least strength 8. But the path strength from p to a is at most 8, too, so if p beats b, then p doesn't beat a.

4.1.2 Manipulation Thresholds

Even if the necessary condition in Proposition 4.2 does not suffice to decide if there is a successful manipulation, it is a starting point to find a criterion: One can try to tighten either side of the inequality, until it can be used to decide the existence of a manipulation. Indeed, we will deal with a broader class of *manipulation thresholds*, to which also the upper bound $S_{D+kI^+}(p,\cdot)$ of Proposition 4.2 belongs. These manipulation thresholds have the property that the threshold value at any candidate exceeds the path strength from that candidate to p for any successful manipulation.

Definition 4.4. Let $I = (V_{\mathrm{N}}, M, p)$ be a manipulation instance with $k = \|M\|$ manipulators.

1. A *manipulation threshold* is a function $\theta : C \setminus \{p\} \to \mathbb{N} \times \mathbb{N}$ such that if there is an unweighted constructive unique winner manipulation, then the value $\theta(c)$ exceeds the strength of any (c,p)-path in an unweighted constructive unique winner manipulation, i.e., for all such manipulations (I, V_{M}) it holds that
$$\theta(c) >_D S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(c,p) \text{ for all } c \in C \setminus \{p\} \ . \tag{4.7}$$
2. A manipulation threshold θ is *tight* if there is an unweighted constructive unique winner manipulation if and only if
$$\theta(c) >_D S_{D(V_{\mathrm{N}})+kI^-}(c,p) \text{ for all } c \in C \setminus \{p\} \ . \tag{4.8}$$

Just as in Proposition 4.2's necessary condition, we can use any manipulation threshold to formulate a necessary condition for the existence of a manipulation. This is a generalization of Proposition 4.2.

Proposition 4.5. *Let $I = (V_{\mathrm{N}}, M, p)$ be a manipulation instance with $k = \|M\|$ manipulators. Let θ be a manipulation threshold. Let $>_D$ be monotonic.*

If there is an unweighted constructive unique winner manipulation, then
$$\theta(c) >_D S_{D(V_{\mathrm{N}})+kI^-}(c,p) \textit{ for all } c \in C \setminus \{p\} \ .$$

Proof. Suppose there is a manipulation (I, V_{M}). Then,
$$\theta(c) >_D S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(c,p) \gtrsim_D S_{D(V_{\mathrm{N}})+kI^-}(c,p) \text{ for all } c \in C \setminus \{p\} \ . \qquad \square$$

Example 4.4.

1. $S_{D+kI^+}(p,\cdot)$ is a manipulation threshold, as for any manipulation (I, V_{M}) we know by Proposition 4.1 that
$$S_{D+kI^+}(p,c) \gtrsim_D S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(p,c) >_D S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(c,p) \text{ for all } c \in C \setminus \{p\} \ .$$

2. $\theta : c \mapsto \max_D\{(D + kI^+)_{ab} | a, b \in C\}$ is also a threshold, because $S_{D+kI^+}(p, c) \lesssim_D \max_D\{(D + kI^+)_{ab} | a, b \in C\}$ for all candidates $c \in C \setminus \{p\}$.

3. A tight threshold is

$$\theta : c \mapsto \begin{cases} S_{D+kI^+}(p, c) & \text{there is a manipulation} \\ S_{D+kI^-}(c, p) & \text{otherwise} \end{cases} .$$

   However, we do not know an efficient way to compute this threshold.

If we know a tight manipulation threshold, we can decide the existence of a manipulation, and if we know whether there is a manipulation, we can give some tight manipulation threshold. Thus, finding some tight manipulation threshold is just as difficult as deciding the existence of a manipulation.

Aim. We are looking for an efficiently computable tight manipulation threshold.

4.1.3 The Restricting Link Rule

The efficiently computable thresholds we covered so far weren't tight. There are two possible ways forward: The first possibility is to somehow find another threshold directly which is tight and efficiently computable; the other possibility is that we develop some rules which allow us to refine our known good thresholds. It is difficult to find an efficiently computable, tight threshold directly, so we will pursue the other possibility: We will try to find rules that we can use to refine manipulation thresholds.

To find out how a threshold can be refined, we study Example 4.3 again. There are 3 links of special interest in this example. There would be an unweighted constructive unique winner manipulation if:

- the strength of the link (p, a) was increased by at least 1, or
- the strength of the link (a, c) was reduced by at least 1, or
- the strength of the link (b, p) was decreased by at least 1.

The following example is a variant of Example 4.3 where the strength of the link (a, c) has been reduced.

Example 4.5. We use margins as our measure of defeat strength. There are two manipulators, and the votes of the non-manipulators resulted in the following pairwise comparison graph:

We test the necessary condition from Proposition 4.2 again. The relevant path strengths after the non-manipulators' vote are shown in Table 4.2.

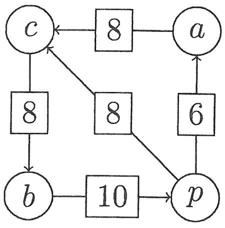

Figure 4.3: Pairwise comparison graph of Example 4.5.

Table 4.2: Path strengths from and to p in Example 4.5.

x	a	b	c
$S_M(p, x)$	6	8	8
$S_M(x, p)$	8	10	8

The path strengths from and to p are the same as in Example 4.3, so Proposition 4.2's necessary condition is again satisfied.

However, in contrast to Example 4.3, there is a way for the manipulators to make p the unique winner: p is the unique winner if both manipulators vote the ranking $p > c > b > a$.

So changing the strength of the link (a, c) in the example can make a manipulation possible; however, the strength of the link is unreflected in the threshold $S_{D+kI^+}(p, \cdot)$. Nevertheless, we can determine the reason why it influences the existence of a manipulation: The link (a, c) induces negative feedback in Example 4.3.

How does this negative feedback in the example work? What happens is that the path strength $S_{D(V_\mathrm{N} \circ V_\mathrm{M})}(c, p)$ for the non-manipulators profile V_N and some potential manipulating profile V_M for the manipulators has to be increased and therefore is fed back to candidate a. The problem arises the moment this feedback exceeds the threshold value $\theta(a)$: The only way to mitigate this effect is to reduce the strength of the link (a, c); the link, however, is so strong that the reduction is too little to prevent the negative feedback. We will call such links that are too strong *restricting links*.

Definition 4.6. Let (V, M, p) be an unweighted manipulation instance with $k = \|M\|$ manipulators. Let $D = D(V)$.

1. A link (a, b) is *ρ-restricting* if and only if $(D + kI^-)_{ab} \gtrsim_D \rho$.
2. A path P is *ρ-restricting* if and only if all of its links are ρ-restricting, i.e., $s_{D+kI^-}(P) \gtrsim_D \rho$.

Remark 4.6. In particular, if the defeat strength measure of choice is the margins measure, a ρ-restricting link is a link (a, b) with $M_{ab} - k \geq \rho$ and a ρ-restricting path is a path P with $s_M(P) - k \geq \rho$.

If we know about this type of negative feedback, we can exploit the mechanism to define a rule that allows us to refine a known manipulation threshold.

Restricting Link rule. Let $s \in C \setminus \{p\}$ be the source candidate and $t \in C \setminus \{p\}$ be the target candidate. If $\theta(s) <_D \theta(t)$ and the link (s, t) is $\theta(s)$-restricting, assign the new threshold value $\theta(s)$ to the candidate t.

Proposition 4.7. *Let* $I = (V_\mathrm{N}, M, p)$ *be an unweighted manipulation instance. Let the defeat strength be monotonic.*

Given a manipulation threshold θ*, an application of the Restricting Link rule produces a manipulation threshold.*

Proof. Let $s, t \in C$ be source and target of the Restricting Link rule application and $D = D(V_\mathrm{N})$ be the defeat strength pairwise matrix. The result of the application is the manipulation threshold

$$\theta'(c) = \begin{cases} \theta(c) & \text{if } c \in C \setminus \{p, t\} \\ \theta(s) & \text{if } c = t \end{cases} .$$

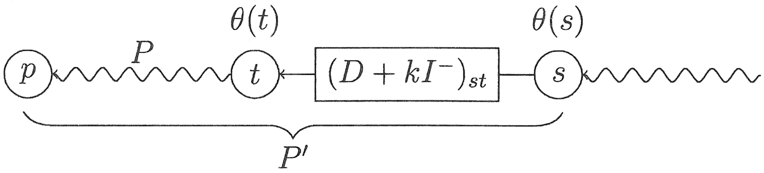

Figure 4.4: A restricting link and the induced negative feedback.

We show the claim by contraposition. Suppose θ' is not a manipulation threshold. Then, there is a successful manipulation (I, V_M) with $\theta'(c) \lesssim_D S_{D(V_\mathrm{N} \circ V_\mathrm{M})}(c, p)$ for a candidate $c \in C \setminus \{p\}$. If $c \neq t$, then θ is obviously no manipulation threshold, for $\theta'(c) = \theta(c)$. Figure 4.4 visualizes the situation for $c = t$. There is a (t, p)-path $P = (c_0, ..., c_\ell)$ in the manipulation with $s_{D(V_\mathrm{N} \circ V_\mathrm{M})}(P) \gtrsim_D \theta'(t) = \theta(s)$. Therefore, the path $P' = (s, c_0, ..., c_\ell)$ is an (s, p)-path with strength

$$\begin{aligned} s_{D(V_\mathrm{N} \circ V_\mathrm{M})}(P') &= \min\nolimits_D\{s_{D(V_\mathrm{N} \circ V_\mathrm{M})}(P), D(V_\mathrm{N} \circ V_\mathrm{M})_{st}\} \\ &\gtrsim_D \min\nolimits_D\{\theta'(t), (D(V_\mathrm{N}) + kI^-)_{st}\} \\ &\gtrsim_D \theta(s) . \end{aligned}$$

Hence, θ is no manipulation threshold. □

We get a tighter manipulation threshold if we apply this rule. Therefore, we refine our original threshold from Proposition 4.1 and apply the Restricting Link rule as often as possible. A sequence of applications of the Restricting Link rule induces a set of restricting paths; consequently, we will define the new threshold in terms of restricting paths. This yields the new threshold θ_1:

$$\begin{aligned}\theta_1(c) &:= \min(\{S_{D+kI^+}(p,c)\} \cup \{S_{D+kI^+}(p,x) \mid \\ &\text{there is a } S_{D+kI^+}(p,x)\text{-restricting } (x,c)\text{-path that doesn't traverse } p\})\ .^1 \end{aligned} \tag{4.9}$$

Example 4.7 (continued from Example 4.3). We continue Example 4.3 and calculate the threshold θ_1. The Restricting Link rule can only be applied once: The candidates a and c are connected by the restricting link (a, c). Therefore, we update candidate c's threshold and get θ_1 as shown in Table 4.3.

Table 4.3: Manipulation threshold θ_1 in Example 4.7.

x	a	b	c
$\theta_1(x)$	8	10	8

In Table 4.1, the path strength from and to p in the example is shown. If we compare the minimum path strength to p (i.e., we subtract 2 from the path strength to p) to the threshold values, we discover that all threshold values are strictly greater than the minimum path strengths to p. However, we already concluded that there is no unweighted constructive unique winner manipulation in Example 4.3. Hence, the threshold is not tight.

4.1.4 The Incoming Paths Rule

Even though the Restricting Link rule refined the manipulation threshold, the rule's application alone did not suffice to make the threshold tight. We need at least one additional rule to refine the thresholds. For this, we consider a candidate c with currently known threshold value $\theta(c)$. If there are manipulations that exhaust the bound $\theta(c)$ on the strength of the strongest (c, p)-path, then there must be a simple (p, c)-path that can supply that much strength from p to c. This path must have the following properties:

- It must be possible for this path to potentially have a strength of at least $\theta(c)$.

---

[1] p is never the source or the target of an application of the Restricting Link rule. As a consequence, a sequence of applications of the Restricting Link rule can never cross over p, so we exclude all restricting paths that contain p.

- All intermediate candidates on the path must have a threshold value of at least $\theta(c)$, too.

If either of these properties aren't fulfilled for the path, the path is unsuitable for the supply of the maximum strength allowed by $\theta(c)$, otherwise θ wouldn't be a manipulation threshold. If there is no such path, then the threshold value $\theta(c)$ cannot be exhausted by any manipulation. Consequently, the threshold value $\theta(c)$ is an overestimation and we can reduce the threshold value until there exists a path with these properties again. This rule is the Incoming Paths rule.

Incoming Paths rule. If for all incoming (p, a)-paths P of a candidate $a \in C \setminus \{p\}$ with at least strength $s_{D+kI^-}(P) \gtrsim_D \theta(a)$ there is an intermediate candidate b with $\theta(b) <_D \theta(a)$, reduce a's threshold value just until the premise becomes false, i.e., update a's threshold value to

$$\max_D\{\min_D\{s_{D+kI^+}(P'), \theta(c_1), ..., \theta(c_{\ell-1})\}\mid \\ P' = (c_0, ..., c_\ell) \text{ is a simple } (p, a)\text{-path }\}\ . \quad (4.10)$$

Proposition 4.8. *Let $I = (V_{\mathrm{N}}, M, p)$ be an unweighted manipulation instance. Let the defeat strength be monotonic.*

Given a manipulation threshold θ, an application of the Incoming Paths rule produces a manipulation threshold.

Proof. Let $a \in C$ be the candidate to whom the rule was applied. The result of the rule application is a threshold θ' with $\theta'(c) = \theta(c)$ for all $c \in C \setminus \{p, a\}$ and

$$\theta'(a) = \max_D\{\min_D\{s_{D+kI^+}(P'), \theta(c_1), ..., \theta(c_{\ell-1})\}\mid \\ P' = (c_0, ..., c_\ell) \text{ is a simple } (p, a)\text{-path}\}\ .$$

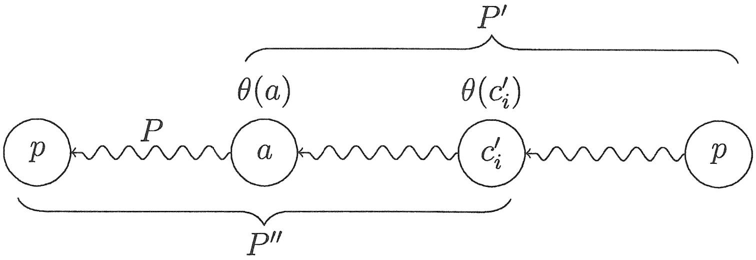

Figure 4.5: The incoming path of a supplying a with the needed strength.

We show the claim by contraposition. Suppose θ' isn't a manipulation threshold, i.e., there is a successful manipulation (I, V_{M}) such that there is a (c, p)-path

$P = (c_0, \ldots, c_\ell)$ with $\theta'(c) \lesssim_D s_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(P)$ for some candidate $c \in C \setminus \{p\}$. If $c \neq a$, then θ is obviously no manipulation threshold, for $\theta(c) = \theta'(c)$. So consider the situation if $c = a$, as visualized in Fig 4.5. Because it is a successful manipulation, there exists a simple (p, a)-path $P' = (c'_0, \ldots, c'_j)$ with $s_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(P') >_D S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(a, p)$. Since $\theta'(a) <_D \theta(a)$ because of the rule's construction and $s_{D(V_{\mathrm{N}})+kI^+}(P') >_D \theta'(a)$, there must be an intermediate candidate c'_i on the path P' with $\theta(c'_i) \lesssim_D \theta'(a) = \theta(a)$. Then, the combined (c_i, p)-path $P'' = (c'_i, \ldots, c'_j = c_0, \ldots, c_\ell)$ has strength

$$
\begin{aligned}
s_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(P'') &\gtrsim_D \min_D\{s_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(P'), s_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(P)\} \\
&\gtrsim_D S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(a, p) \\
&\gtrsim_D \theta(c'_i)
\end{aligned}
$$

Hence, θ is no manipulation threshold. □

Remark 4.8. The test whether the rule is applicable is complicated, costly and is not simpler than computing the new threshold value itself. For that reason, we will never do the test explicitly, but instead candidate a's threshold value is always updated to

$$
\max_D\{\min_D\{s_{D+kI^+}(P'), \theta(c_1), \ldots, \theta(c_\ell)\} | P' = (c_0, \ldots, c_\ell) \text{ is } (p, a)\text{-path } \} :
$$

If the term yields $\theta(a)$, the premise wasn't satisfied; if the term yields something smaller, the rule was applicable and applied. Note that calculating the expression is a single-source single-destination strongest path problem: It is the strength of the strongest (p, a)-path in the weighted digraph $G = (C, E; w)$ with edge set $E = C \times (C \setminus \{p\})$ and edge weights $w : E \to \mathbb{N} \times \mathbb{N} : (a, b) \mapsto \min_D\{(D + kI^+)_{ab}, \theta(b)\}$.

If we apply the Incoming Paths rule as often as possible to the example threshold θ_1, we gain a new threshold θ_2:

$$
\theta_2(c) := \max\{\min\{s_{D+kI^+}(P), \theta_1(c_1), \ldots, \theta_1(c_\ell)\} | P = (c_0, \ldots, c_\ell) \text{ is } (p, c)\text{-path}\}
$$

Example 4.9 (continued from Example 4.3 and 4.7). Continuing the examples, we now calculate the threshold θ_2 using the threshold θ_1 in Table 4.3. Table 4.4 exhibits the specific values after the threshold has been calculated.

Table 4.4: Manipulation threshold θ_2 in Example 4.9.

x	a	b	c
$\theta_2(x)$	8	8	8

If we compare this to the minimum path strength from any candidate to p, i.e., we compare this to the path strength to p shown in Table 4.1 from which we subtract 2, we see that the minimum path strength from b to p and $\theta_2(b)$ are the same. Therefore, we know that there is no manipulation in this example.

However, even θ_2 is not yet a tight manipulation threshold. Example 4.10 presents a case where there is a manipulation, even though there shouldn't be one if θ_2 were tight.

Example 4.10. We use margins again as our measure of defeat strength. There are two manipulators, and the votes of the non-manipulators resulted in the following pairwise comparison graph:

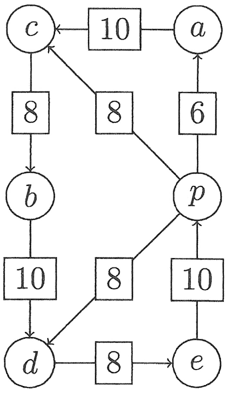

Figure 4.6: Pairwise comparison graph of Example 4.10

Computing the path strengths yields Table 4.5.

Table 4.5: Path strengths from and to p in Example 4.10.

x	a	b	c	d	e
$S_M(p,x)$	6	8	8	8	8
$S_M(x,p)$	8	8	8	8	10

Afterwards, we can calculate the thresholds θ_1 and θ_2, as shown in Table 4.6 and Table 4.7.

Table 4.6: Manipulation threshold θ_1 in Example 4.10.

x	a	b	c	d	e
$\theta_1(x)$	8	10	8	10	10

Table 4.7: Manipulation threshold θ_2 in Example 4.10.

c	a	b	c	d	e
$\theta_2(x)$	8	8	8	10	10

If the threshold θ_2 is compared with the minimum path strength to p after the manipulators' vote, we see that $\theta_2(c)$ is always strictly less than $S_M(c,p)-2$. Thus, θ_2 does not eliminate the possibility that there is a successful manipulation.

However, there is a convincing argument that there cannot be a manipulation: Suppose there is a successful manipulation. Then, there must be a path from p to e of at least strength 9. Thus, there is a path from d to p of at least strength 8. Consequently, there is also a path from b to p of at least strength 8, so there must also be a path from p to b of at least strength 9. If there is such a path, then there is also a path from c to p of at least strength 8, and, by extension, from a to p. There can only be a path of strength 8 from p to a, though, which is a contradiction.

4.1.5 The Beat Threshold

Even though the manipulation threshold θ_2 is an improvement compared to the simple thresholds we started out with, it is still not tight, so it cannot decide the existence of a manipulation in the general case. However, we found two rules with which we can refine a manipulation threshold. In the case of θ_2, we only applied each rule at best a linear number of times, at worst only once. Next, we are going to apply the rules as often as possible until they can no longer be used. This leads to Algorithm 1 and a new manipulation threshold.

Algorithm 1: General Algorithm for Beat Threshold Computation

```
Input: Finite, non-empty set of candidates $C$
       Defeat strength pairwise matrix $D \in (\mathbb{N} \times \mathbb{N})^{C \times C}$
       Number of manipulators $k$
       A candidate $p \in C$ who is preferred by the manipulators
Output: The beat threshold $\mathrm{bt} : C \setminus \{p\} \to \mathbb{N} \times \mathbb{N}$
begin
    $\mathrm{bt} \equiv \max_D\{(D + kI^+)_{ab} | a, b \in C\}$;
    while the Restricting Link rule or the Incoming Paths rule is applicable to bt do
        apply the rule;
    return bt;
```

Proposition 4.9. *Algorithm 1 terminates.*

Proof. The algorithm initializes bt with the maximum defeat strength of a link possible after the manipulators' vote. If a rule is applied, it assigns a new value to one candidate's $\mathrm{bt}(c)$. This new value is either a value that is currently assigned to another candidate c', or it is the strength of a path, which must be the strength of a link. Therefore, all possible intermediate values and values at the end assigned to bt must originate from the matrix $D + kI^+$. If all values stem from $D + kI^+$, there can be at most n^2 different values.

If a rule is applied, it reduces the threshold of one candidate without touching the threshold of the others. Consequently, there can be at most $n^2 - 1$ rule applications reducing the threshold value for one specific candidate. Since there are $n - 1$ different candidates with an assigned threshold, there are at most $(n^2 - 1)(n - 1)$ rule applications in total. Hence, the algorithm terminates. □

The result of the algorithm is the *beat threshold* bt. In the computation of the beat threshold, both rules we established are exhaustively applied, so this is the smallest possible threshold we can get using these two rules. In Lemma 4.10, we formally establish that the beat threshold is a manipulation threshold.

Lemma 4.10. *Let* $I = (V_{\mathrm{N}}, M, p)$ *be an unweighted manipulation instance. Let the defeat strength be monotonic.*

The beat threshold is a manipulation threshold.

Proof. Let (I, V_{M}) be any unweighted constructive unique winner manipulation. Let bt_k be the value of the beat threshold estimate after the k-th iteration of Algorithm 1.

We show the by induction on the number of iterations k in Algorithm 1 that each iteration produces a manipulation threshold. Consequently, the beat threshold is a manipulation threshold, too.

- Basis: $k = 0$: Since the defeat strength measure is monotonic, we know that a link's strength for the manipulation's voting profile is at most as strong as if all manipulators supported the link, i.e., $D(V_{\mathrm{N}} \circ V_{\mathrm{M}})_{ab} \precsim_D (D + kI^+)_{ab}$ for all candidates $a, b \in C$. Therefore, the strength of no path can exceed the maximum strength of some link in matrix $D + kI^+$, so

  $$\underbrace{S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(c,p) <_D S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(p,c)}_{\text{successful manipulation}} \precsim_D \max\nolimits_D\{(D+kI^+)_{ab} | a,b \in C\} = \mathrm{bt}_0(c)\ .$$

  Hence, bt_0 is a manipulation threshold.

- Inductive step: $k > 0$: In each iteration we apply either the Restricting Link rule or the Incoming Paths rule. By Proposition 4.7 and Proposition 4.8, they produce a manipulation threshold, for they are applied to a manipulation threshold. □

The beat threshold, as it is defined by the algorithm, can be computed using Algorithm 1, but Algorithm 1 does not provide an efficient strategy in which order the two different rules should be applied.

As already mentioned at the Incoming Paths rule's introduction, the computational problem underlying the rule is in principle a single-source single-destination strongest path problem. However, the algorithms used to solve the single-source single-destination strongest path problem are single-source strongest path algorithms. Therefore, a good strategy is to handle the Incoming Paths rule en bloc and to apply it to all candidates at once.

Algorithm 2 is a variant of Algorithm 1 in which each rule is applied en bloc, switching to the other rule if the current one is no longer applicable. If the Restricting Link rule is not applicable even once after the Incoming Paths rule was exhaustively applied, we can stop, for both rules are not applicable.

Algorithm 2: Algorithm Variant for Beat Threshold Computation

```
Input: Finite, non-empty set of candidates $C$
       Defeat strength pairwise matrix $D \in (\mathbb{N} \times \mathbb{N})^{C \times C}$
       Number of manipulators $k$
       A candidate $p \in C$ who is preferred by the manipulators
Output: The beat threshold $\mathrm{bt} : C \setminus \{p\} \to \mathbb{N} \times \mathbb{N}$
begin
    $\mathrm{bt} \equiv \max_D\{(D + kI^+)_{ab} | a, b \in C\}$;
    repeat
        while the Incoming Paths rule is applicable to bt do
            apply the rule;
        stable ← true;
        while the Restricting Link rule is applicable to bt do
            stable ← false;
            apply the rule;
    until stable;
    return bt;
```

In order to analyze the correctness and the runtime behaviour of the following algorithms, we introduce the concept of a stable candidate.

Definition 4.11. A candidate $c \in C \setminus \{p\}$ is *stable* with respect to a manipulation threshold θ if and only if

1. c is not the source of a potential application of the Restricting Link rule to θ, and
2. there is no candidate $c' \in C \setminus \{p\}$ with $\theta(c') < \theta(c)$ that is the source of a potential application of the Restricting Link rule to θ.

Lemma 4.12. *Algorithm 2's outer repeat loop does at most $\lceil \frac{n}{2} \rceil$ iterations.*

Proof. We take snapshots of the bt variable in Algorithm 2 in each iteration of the repeat loop. $\widehat{\mathrm{bt}}_k$ is the variable's value in the kth iteration just before the first while loop is executed and bt_k is the variable's value in the kth iteration just before the second while loop is executed.

Let Q_k be the set of candidates which are stable w.r.t. bt_k. We prove for all iterations that the number of stable candidates increases by at least two as long as there are non-stable candidates, i.e.,

$$\|Q_{k+1}\| \geq 2 + \|Q_k\| \text{ if } Q_k \subset C \setminus \{p\}\ . \tag{4.11}$$

Suppose there are candidates that aren't stable w.r.t. bt_k. Consequently, there is a non-stable candidate s with minimal threshold value $\mathrm{bt}_k(s)$. All potential Restricting Link rule applications have a source candidate with a threshold value of at least $\mathrm{bt}_k(s)$, so if the threshold value of a candidate is reduced, it cannot fall below $\mathrm{bt}_k(s)$. Thus,

$$\min_D\{\mathrm{bt}_k(s), \mathrm{bt}_k(c)\} \lesssim_D \widehat{\mathrm{bt}}_{k+1}(c) \lesssim_D \mathrm{bt}_k(c) \text{ for all } c \in C \setminus \{p\}. \tag{4.12}$$

For each candidate $c \in C \setminus \{p\}$, there is a simple (p, c)-path $P = (c_0, \ldots, c_\ell)$ with $s_{D+kI^+}(P) \gtrsim_D \mathrm{bt}_k(c)$ and $\mathrm{bt}_k(c_i) \gtrsim_D \mathrm{bt}_k(c)$ for $c_1, \ldots, c_\ell$, for the Incoming Paths rule is not applicable. For all of these candidates, (4.12) holds, so $\widehat{\mathrm{bt}}_{k+1}(c_i) \gtrsim_D \min_D\{\mathrm{bt}_k(c), \mathrm{bt}_k(s)\}$. Therefore, all these candidates must still have a threshold value of at least $\min_D\{\mathrm{bt}_k(c), \mathrm{bt}_k(s)\}$ after the Incoming Paths rule has been applied, for each sub-path of P is at least that strong and all of the intermediate candidates of that sub-path also have at least that strength. Consequently, the Incoming Paths rule cannot reduce the threshold value below $\min_D\{\mathrm{bt}_k(c), \mathrm{bt}_k(s)\}$. Thus,

$$\min_D\{\mathrm{bt}_k(s), \mathrm{bt}_k(c)\} \lesssim_D \mathrm{bt}_{k+1}(c) \lesssim_D \mathrm{bt}_k(c) \text{ for all } c \in C \setminus \{p\}. \tag{4.13}$$

Consider a candidate $c \in C \setminus \{p\}$ with $\widehat{\mathrm{bt}}_{k+1}(c) \lesssim_D \mathrm{bt}_k(s)$. Equation (4.13) tells us that the Incoming Paths rule does not change his threshold value, so $\mathrm{bt}_{k+1}(c) \sim_D \widehat{\mathrm{bt}}_{k+1}(c)$. This has consequences for the applicability of the Restricting Link rule: If candidate c is source of a potential application of the Restricting Link rule to $\mathrm{bt}_{k+1}(c)$, then it would also be a source for a potential application of the Restricting Link rule to $\widehat{\mathrm{bt}}_{k+1}(c)$, for the threshold value of any candidate never increases. By construction, the Restricting Link rule can't be applied to $\widehat{\mathrm{bt}}_{k+1}$, so c cannot be the source of any potential application of a Restricting Link rule. Moreover, all candidates c' with $\widehat{\mathrm{bt}}_{k+1}(c') >_D \mathrm{bt}_k(s)$ have at least a threshold value of $\mathrm{bt}_{k+1}(c') \gtrsim_D \mathrm{bt}_k(s)$, so they do not influence whether the candidate c is stable. Consequently, all candidates c with $\widehat{\mathrm{bt}}_{k+1}(c) \lesssim_D \mathrm{bt}_k(s)$ are stable w.r.t. bt_{k+1}.

All candidates that are stable w.r.t. bt_k have a threshold value of at most $\mathrm{bt}_k(s)$, so they are also stable w.r.t. bt_{k+1}. Additionally, s and a target t of s's potential rule application become stable, too: For s, the threshold value after all applications of the Restricting Link rule cannot have increased, so $\widehat{\mathrm{bt}}_{k+1}(s) \lesssim_D \mathrm{bt}_k(s)$. The Restricting Link rule must have been applied to t so that the Restricting Link rule was no longer applicable from source s to t, so we can also conclude that $\widehat{\mathrm{bt}}_{k+1}(t) \lesssim_D \mathrm{bt}_k(s)$. As we have seen, such candidates are stable w.r.t. bt_{k+1}. Since all stable candidates remain stable and we found two additional candidates that become stable, the number of stable candidates increases by at least two.

Consequently, there are at least $\min\{2(k-1), n-1\}$ stable candidates in the kth iteration. Since there are always at least 2 unstable candidates, if some candidate is unstable, all candidates must be stable as soon as the minimum number of stable candidates is at least $n-2$. As the outer loop is left when all candidates are stable, the outer loop is left at the latest in the first iteration k with

$$2(k-1) \geq n-2 \iff k \geq \frac{n}{2} \, .$$

Hence, Algorithm 2's outer loop has at most $\lceil \frac{n}{2} \rceil$ iterations. □

Finally, we only need to find an efficient implementation of the inner loops of Algorithm 2. For the first loop, we can use the equivalence of the Incoming Paths rule to the strongest path problem. Algorithm 3 efficiently implements both loops.

Lemma 4.13. *Algorithm 3 computes the beat threshold in $\mathcal{O}(n^3)$ time and $\mathcal{O}(n)$ space.*

Proof. We show that this algorithm is an implementation of Algorithm 2. For that, we have to show that the first statement of the outer loop applies the Incoming Paths rule as often as possible and that the remaining loop body applies the Restricting Link rule exhaustively.

Algorithm 3: Algorithm for Efficient Computation of the Beat Threshold

```
Input: Finite, non-empty set of candidates $C$
       Defeat strength pairwise matrix $D \in (\mathbb{N} \times \mathbb{N})^{C \times C}$
       Number of manipulators $k$
       A candidate $p \in C$ who is preferred by the manipulators
Data: Priority queue $Q$
Output: Beat threshold $\mathrm{bt} : C \setminus \{p\} \to \mathbb{N} \times \mathbb{N}$
begin
    $\mathrm{bt} \equiv \max_D\{(D + kI^+)_{ab} | a, b \in C\}$;
    repeat
        $\mathrm{bt}'(\cdot) \leftarrow$ compute path strength from $p$ in weighted digraph
        $G = (C, E; \lambda)$ with edge set $E = C \times (C \setminus \{p\})$ and weights
        $\lambda : E \to \mathbb{N} \times \mathbb{N} : (a, b) \mapsto \min_D\{(D + kI^+)_{ab}, \mathrm{bt}(b)\}$;
        foreach $c \in C \setminus \{p\}$ do
            $Q$.insert($c$, $\mathrm{bt}'(c)$);
        stable $\leftarrow$ true;
        while $Q \neq \emptyset$ do
            $c \leftarrow Q$.extractMin();
            foreach $a \in C \setminus \{c\}$ do
                if $\mathrm{bt}'(a) >_D \mathrm{bt}'(c)$ and $\mathrm{bt}'(c) \lesssim_D (D + kI^-)_{ca}$ then
                    $\mathrm{bt}'(a) \leftarrow \mathrm{bt}'(c)$;
                    $Q$.updateKey($a$, $\mathrm{bt}'(a)$);
                    stable $\leftarrow$ false;
        $\mathrm{bt} \leftarrow \mathrm{bt}'$;
    until stable;
    return bt;
```

In the case of the first statement, we compute the strength of the strongest (p, c)-path for all candidates $c \in C \setminus \{p\}$ in the graph G. Let bt be the value of the bt variable before the statement. Consider a (p, c)-path $P = (c_0, ..., c_\ell)$. The strength of the path in graph G is

$$\begin{aligned} s_\lambda(P) &= \min_D\{\lambda(c_{i-1}, c_i) | i \in \{1, ..., \ell\}\} \\ &= \min_D\{\min_D\{(D + kI^+)_{c_{i-1}c_i}, \mathrm{bt}(c_i)\} | i \in \{1, ..., \ell\}\} \\ &= \min_D\{s_{D+kI^+}(P), \mathrm{bt}(c_1), ..., \mathrm{bt}(c_\ell)\} \ . \end{aligned}$$

Without loss of generality, let P be the path with maximum strength. Then, $\mathrm{bt}'(c)$ takes the value $s_\lambda(P)$. If $\mathrm{bt}'(c) \sim \mathrm{bt}(c)$, then the Incoming Paths rule was not applicable, so the threshold value remains the same. If $\mathrm{bt}'(c) < \mathrm{bt}(c)$, the Incoming Paths rule was applied. Moreover, the Incoming Paths rule cannot be applied again to candidate c: For all intermediate candidates $c_i, i \in \{1, ..., \ell\}$ on the path, the subpath $P' = (c_0, ..., c_i)$ has at least strength $s_\lambda(P') \gtrsim_D \mathrm{bt}'(c)$. Consequently, $\mathrm{bt}'(c_i) \gtrsim_D \mathrm{bt}'(c)$, so the Incoming Paths rule cannot be applied again to c.

Next, we have to show that the loops following the first statement are consistent with exhaustive applications of the Restricting Link rule. Let $\widehat{\mathrm{bt}}_k$ be the state of the bt' variable and Q_k be the set of elements in the priority queue after the kth iteration of the while loop. First, it is clear that the beat threshold is always only changed in accordance with the Restricting Link rule. Therefore, we only have to show that the Restricting Link rule isn't applicable at the end.

We claim the following invariant of the while loop: After the kth iteration, all candidates $c \in C \setminus (Q_k \cup \{p\})$ that are not in the queue are stable w.r.t. $\widehat{\mathrm{bt}}_k$.

- Basis: $k = 0$: Obviously, $Q_0 = C \setminus \{p\}$. As a consequence, there are no candidates for whom we have to show the invariant.

- Inductive step: $n > k > 0$: First, the algorithm determines the candidate c with minimum threshold value $\widehat{\mathrm{bt}}_{k-1}(c)$ in the queue Q_{k-1}. Afterwards, we attempt to apply the Restricting Link rule to all potential targets. Consequently, c is no longer the source of any potential application of the Restricting Link rule. Furthermore, all targets either keep their threshold value or get the threshold value $\widehat{\mathrm{bt}}_{k-1}(c)$. Since c was not stable, all stable candidates w.r.t. $\widehat{\mathrm{bt}}_{k-1}$ must have a threshold value of at most $\widehat{\mathrm{bt}}_{k-1}(c)$. Thus, they are also stable w.r.t. $\widehat{\mathrm{bt}}_{k-1}$. By induction, all candidates not in queue Q_{k-1} are stable w.r.t. $\widehat{\mathrm{bt}}_{k-1}$. Additionally, there is no source of a potential application of the Restricting Link rule with a threshold value smaller than $\widehat{\mathrm{bt}}_{k-1}(c)$, so c is now stable, too. Hence, all candidates that are not in Q_k are stable w.r.t. $\widehat{\mathrm{bt}}_k$.

After $n - 1$ iterations, Q is empty; as a consequence of the invariant, none of the candidates $c \in C \setminus \{p\}$ can be the source of a potential application of the Restricting Link rule. Thus, the Restricting Link rule is not applicable to $\widehat{\mathrm{bt}}_{n-1}$.

Hence, the body of the outer loop of Algorithm 3 complies with the body of the outer loop of Algorithm 2, so the algorithm is correct.

The body of the outer loop can be implemented in $\mathcal{O}(n^2)$ time: The strongest path in the path G from the source p can be computed using the Algorithm of Dijkstra adapted to the single-source strongest path problem. This algorithm can be implemented in $\mathcal{O}(n^2)$ time and using $\mathcal{O}(n)$ space [Dij59]. The first inner loop can be implemented in $\mathcal{O}(n)$ time, while the second one needs $\mathcal{O}(n^2)$ time. Additional $\mathcal{O}(n)$ space are needed for the priority queue and the variables bt and bt'. Since the number of iterations of the outer loop is at most $\lceil \frac{n}{2} \rceil$ by Lemma 4.12, the algorithm runs in $\mathcal{O}(n^3)$ time and $\mathcal{O}(n)$ space. □

4.1.6 Computing a Manipulation

Whilst we established in the previous subsection what the beat threshold is and how the beat threshold can be efficiently computed, we barely touched whether the beat threshold has significance in our quest for an efficient way to compute a manipulation: We only demonstrated that the beat threshold is a manipulation threshold, so we can use it as part of a necessary condition for the existence of a manipulation.

Here, Lemma 4.14 is central in our efforts: It demonstrates a sufficient condition, based on the beat threshold, for the existence of the manipulation that even works when the defeat strength is not monotonic.

Lemma 4.14. *Let $I = (V_{\mathrm{N}}, M, p)$ be an unweighted manipulation instance with $k = \|M\|$ manipulators. Let $D = D(V_{\mathrm{N}})$.*

There is an unweighted constructive unique winner manipulation, if

$$(D + kI^-)_{cp} <_D \mathrm{bt}(c) \text{ for all } c \in C \setminus \{p\} \ . \tag{4.14}$$

Proof. Let $G = (C, E)$ be a directed graph with edge set

$$\begin{aligned} E :=& \{(p,a) | a \in C \setminus \{p\}, (D + kI^+)_{pa} \gtrsim_D \mathrm{bt}(a)\} \cup \\ & \{(a,b) | a, b \in C \setminus \{p\}, \min_D\{\mathrm{bt}(a), (D + kI^+)_{ab}\} \gtrsim_D \mathrm{bt}(b)\} \ . \end{aligned}$$

For each candidate $c \in C \setminus \{p\}$ there is a (p, c)-path $P = (c_0, ..., c_\ell)$ such that

$$\min_D\{s_{D+kI^+}(P), \mathrm{bt}(c_1), ..., \mathrm{bt}(c_{\ell-1})\} \gtrsim_D \mathrm{bt}(c) \ ,$$

for the Incoming Paths rule is not applicable to the beat threshold.

An inductive argument over all non-equivalent beat threshold values then implies that there is then also a simple (p, c)-path $P' = (c'_0, ..., c'_j)$ such that $(D + kI^+)_{pc'_1} \gtrsim_D \mathrm{bt}(c'_1)$ and

$$\min_D\{(D + kI^+)_{c'_{i-1}c'_i}, \mathrm{bt}(c'_{i-1})\} \gtrsim_D \mathrm{bt}(c'_i) \text{ for all } i \in \{2, ..., j\} \ .$$

Therefore, all traversed edges of the path P' are also edges in graph G. Thus, any candidate c is reachable from p in G, so there exists an arborescence of G rooted at p.

Let $T \subseteq G$ be a spanning arborescence of G rooted at p. Let R a complete ranking with the properties that

$$\begin{aligned} (a,b) \in E(T) &\implies a >_R b \text{ for all } a, b \in C \text{ and} \\ \mathrm{bt}(a) >_D \mathrm{bt}(b) &\implies a >_R b \text{ for all } a, b \in C \setminus \{p\} \ . \end{aligned}$$

Such a complete ranking R exists, for T is acyclic and never contains an edge $(a, b) \in E(T)$ where $\mathrm{bt}(b) >_D \mathrm{bt}(a)$, by the construction of the directed graph G.

Let $V_{\mathrm{M}} = (R, ..., R) \in \mathcal{T}^k$ be a manipulators' voting profile. We show that (I, V_{M}) is an unweighted constructive unique winner manipulation if (4.14) holds:

Let $c \in C \setminus \{p\}$ be any candidate.

- Since T is an arborescence rooted at p, c is reachable from p. Thus, there is a (p, c)-path $P = (c_0, ..., c_\ell)$ in T. We know that $(D + kI^+)_{c_{i-1}c_i} \gtrsim_D \mathrm{bt}(c_i)$ for all $i \in \{1, ..., \ell\}$, since this is a necessary condition for the inclusion of the edge in the graph G. From that, we can infer that $(D + kI^+)_{c_{i-1}c_i} \gtrsim_D \mathrm{bt}(c)$, for the second condition for the inclusion of an edge, namely $\mathrm{bt}(c_{i-1}) \gtrsim_D \mathrm{bt}(c_i)$ for all $i \in \{2, ..., \ell\}$, especially implies by a simply inductive argument that $\mathrm{bt}(c_i) \gtrsim_D \mathrm{bt}(c)$ for all $i \in \{1, ..., \ell\}$.

  Because R is a topological ordering of T, all candidates that are part of the path are ordered in the order $c_0 >_R ... >_R c_\ell$. Thus, we know for the strength of the path P for the manipulators' voting profile V_{M} that

$$\begin{aligned} s_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(P) &= \min{}_D\{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})_{c_{i-1}c_i} | i \in \{1, ..., \ell\}\} \\ &= \min{}_D\{(D + kI^+)_{c_{i-1}c_i} | i \in \{1, ..., \ell\}\} \quad (\text{since } c_{i-1} >_R c_i\ ) \\ &\gtrsim_D \mathrm{bt}(c) \qquad\qquad\qquad (\text{since } (D + kI^+)_{c_{i-1}c_i} \gtrsim_D \mathrm{bt}(c)\ ). \end{aligned}$$

  Hence,

$$S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(p, c) \gtrsim_D \mathrm{bt}(c) \text{ for all } c \in C \setminus \{p\}\ . \tag{4.15}$$

- Let $P = (c_0, ..., c_\ell)$ be any simple (c, p)-path. Depending on the beat threshold values of the candidates in the path, there are two cases:

Case 1: The beat threshold of no intermediate candidate on the path exceeds $\mathrm{bt}(c)$, i.e., $\mathrm{bt}(c_i) \lesssim_D \mathrm{bt}(c)$ for all $i \in \{0, ..., \ell - 1\}$. Since $c_{\ell-1}$ is reachable from p in T and R is a topological ordering of T, p is ranked above $c_{\ell-1}$ in R. Thus, we can infer that

$$\begin{aligned} s_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(P) &= \min{}_D\{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})_{c_{i-1}c_i} | i \in \{1, ..., \ell\}\} \\ &\lesssim_D D(V_{\mathrm{N}} \circ V_{\mathrm{M}})_{c_{\ell-1}p} \\ &= (D + kI^-)_{c_{\ell-1}p} \qquad (\text{since } p >_R c_{\ell-1}\ ) \\ &<_D \mathrm{bt}(c_{\ell-1}) \qquad\quad (\text{by } (4.14)) \\ &\lesssim_D \mathrm{bt}(c)\ . \end{aligned}$$

Case 2: The beat threshold of an intermediate candidate on the path exceeds $\mathrm{bt}(c)$, i.e., $\mathrm{bt}(c_i) >_D \mathrm{bt}(c)$ for some $i \in \{1, ..., \ell - 1\}$. Then, there is a first candidate c_i in the path whose beat threshold exceeds c's, i.e.

such a c_i so that i is minimal. By construction, the candidate before c_i, c_{i-1}, must have a beat threshold that is strictly smaller than c's beat threshold, i.e., $\mathrm{bt}(c_{i-1}) <_D \mathrm{bt}(c_i)$. We know that the Restricting Link rule is no longer applicable to the beat threshold bt, so especially the link (c_{i-1}, c_i) isn't $\mathrm{bt}(c_{i-1})$-restricting. Furthermore, we can tell for the ranking R that $c_i >_R c_{i-1}$, for the beat threshold of c_i is strictly greater than the beat threshold of c_{i-1}. Thus,

$$\begin{aligned}
s_{D(V_\mathrm{N} \circ V_\mathrm{M})}(P) &= \min{}_D\{D(V_\mathrm{N} \circ V_\mathrm{M})_{c_{j-1}c_j} | j \in \{1, \ldots, \ell\}\} \\
&\lesssim_D D(V_\mathrm{N} \circ V_\mathrm{M})_{c_{i-1}c_i} \\
&= (D + kI^-)_{c_{i-1}c_i} \quad \text{(since } c_i >_R c_{i-1} \text{)} \\
&<_D \mathrm{bt}(c_{i-1}) \quad \text{(for } (c_{i-1}, c_i) \text{ isn't } \mathrm{bt}(c_{i-1})\text{-restricting)} \\
&\lesssim_D \mathrm{bt}(c) \ .
\end{aligned}$$

If we combine both cases, we get for the path strength that

$$S_{D(V_\mathrm{N} \circ V_\mathrm{M})}(c, p) <_D \mathrm{bt}(c) \text{ for all } c \in C \setminus \{p\} \ . \tag{4.16}$$

Using the equations (4.15) and (4.16), we find that

$$S_{D(V_\mathrm{N} \circ V_\mathrm{M})}(p, c) \gtrsim_D \mathrm{bt}(c) >_D S_{D(V_\mathrm{N} \circ V_\mathrm{M})}(c, p) \text{ for all } c \in C \setminus \{p\} \ .$$

Hence, we find under the assumption of (4.14) and using Proposition 4.1 that (I, V_M) is an unweighted constructive unique winner manipulation. ☐

If we combine both the necessary and the sufficient condition based on the beat threshold, we can establish that the beat threshold is tight. Thus, we finally found an efficiently computable criterion to decide the existence of a manipulation.

Theorem 4.15. *Let (V, M, p) be an unweighted manipulation instance with $k = \|M\|$ manipulators. Let $D = D(V)$. Let the defeat strength be monotonic.*

1. *The beat threshold is a tight manipulation threshold.*
2. *There is an unweighted constructive unique winner manipulation if and only if*

$$(D + kI^-)_{cp} <_D \mathrm{bt}(c) \textit{ for all } c \in C \setminus \{p\} \ . \tag{4.17}$$

Proof. For all candidates $c \in C \setminus \{p\}$, the link (c, p) forms a (c, p)-path, so $(D + kI^-)_{cp} \lesssim_D S_{D+kI^-}(c, p)$. Both claims in the theorem follow from this relation between the strength of the link and the strongest path together with Proposition 4.5 and Lemma 4.10. ☐

Algorithm 4: Algorithm for Unweighted Constructive Unique Winner Manipulation of the Schulze method

```
Input: Finite, non-empty set of candidates $C$
       Defeat strength pairwise matrix $D \in (\mathbb{N} \times \mathbb{N})^{C \times C}$
       Number of manipulators $k$
       A candidate $p \in C$ who is preferred by the manipulators
Output: A complete ranking $R$ such that $(R, ..., R) \in \mathcal{T}^k$ is the manipulators'
        voting profile for a constructive unique winner manipulation, or
        nothing if there is no manipulation
begin
    $\mathrm{bt}(\cdot) \leftarrow$ compute beat threshold;
    if there ex. $c \in C \setminus \{p\}$ with $(D_{cp} + kI^-) \gtrsim_D \mathrm{bt}(c)$ then
        return no unweighted constructive unique winner manipulation;
    Let $G = (C, E)$ be a directed graph with
        $E := \{(p, a) | a \in C \setminus \{p\} \wedge (D + kI^+)_{pa} \gtrsim_D \mathrm{bt}(a)\} \cup$
        $\{(a, b) | a, b \in C \setminus \{p\} \wedge \min_D\{\mathrm{bt}(a), (D + kI^+)_{ab}\} \gtrsim_D \mathrm{bt}(b)\}$ ;
    Let $T \subseteq G$ be a spanning arborescence rooted at $p$;
    Let $R$ be topological ordering of $T$ such that $\mathrm{bt}(a) >_D \mathrm{bt}(b) \implies a >_R b$
    for all $a, b \in C \setminus \{p\}$;
    return $R$;
```

Lemma 4.14's proof contained a construction that determined a manipulation for any manipulation instance under the assumption of Theorem 4.15's criterion. This construction can also be turned into an algorithm, so that we have finally found a technique to compute a manipulation in polynomial time:

Theorem 4.16. *Let the defeat strength be monotonic. Let (V, M, p) be a manipulation instance with $k = \|M\|$ manipulators and m non-manipulators.*

1. *Algorithm 4 computes an unweighted constructive unique winner manipulation's ballot for all manipulators in $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space, if one exists.*

2. *An unweighted constructive unique winner manipulation can be computed in $\mathcal{O}(n^2(n + m + k))$ time and $\mathcal{O}(kn^2)$ space.*

3. *If there is an unweighted constructive unique winner manipulation, then all manipulators can vote the same complete ranking to manipulate successfully.*

Proof.

1. For the correctness, cf. Lemma 4.14's proof and Theorem 4.15. The beat threshold can be computed in $\mathcal{O}(n^3)$ time (by Lemma 4.13) and $\mathcal{O}(n)$ space. The graph G, its arborescence T and the topological ordering R can also be computed in $\mathcal{O}(n^2)$ time using $\mathcal{O}(n^2)$ space. Therefore, Algorithm 4 runs in $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space.

2. Additionally, we need to compute the Condorcet pairwise matrix, which takes $\mathcal{O}(mn^2)$ time and $\mathcal{O}(n^2)$ space. Moreover, we need to construct the manipulators' voting profile. Depending on the way the profile is represented, we have to make a copy of the ranking for each manipulator, which can take at most $\mathcal{O}(kn^2)$ time and needs at most $\mathcal{O}(kn^2)$ space. Hence, we need at most $\mathcal{O}(n^2(n+m+k))$ time.

3. Algorithm 4 always computes a complete ranking that should be voted by all manipulators. ☐

4.1.7 Complexity

Whilst we showed that we can find an unweighted constructive unique winner manipulation in polynomial time, we did not rigidly determine how hard the UNWEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION decision problem is. This is also of interest because the problem was much simpler for 1 manipulator, when all non-manipulators vote complete rankings (cf. Theorem 4.3). However, as we will see, the test must be more complex in the general case with at least 2 manipulators than in the case for one manipulator: One consequence of the results is that the problems belong to different complexity classes, if we assume that NL $\neq$ P.

Before we start the complexity analysis, we first state an observation that all monotonic defeat strength measures are equivalent to the margins measure for some special voting profiles; especially, they are equivalent if all voters vote complete rankings. Due to this, it suffices to analyze the hardness for the margins measure in all reductions, for all constructed voting profiles satisfy Proposition 4.17's precondition.

Proposition 4.17. *Let the defeat strength be monotonic. Let V be an unweighted $|V|$-voter voting profile with Condorcet pairwise matrix $N = N(V)$ such that there is at most one voter in V that didn't order two different candidates, i.e., $N_{ab} + N_{ba} \geq |V| - 1$ for all $a, b \in C$ with $a \neq b$. Let $D = D(V)$ be the defeat strength pairwise matrix and $M = M(V)$ be the margins pairwise matrix.*

Then, $D_{ab} >_D D_{cd} \iff M_{ab} > M_{cd}$ for all $a, b, c, d \in C$ with $a \neq b$ and $c \neq d$.

Proof. Let $a, b, c, d \in C$ be candidates with $a \neq b$ and $c \neq d$. Without loss of generality, assume that $M_{ab} \geq M_{cd}$. There are the following cases:

1. $M_{ab} = M_{cd}$. Since $N_{ab} - N_{ba}$ is odd if and only if $N_{ab} + N_{ba}$ is odd, both $N_{ab} + N_{ba}$ and $N_{cd} + N_{dc}$ must be either odd or even, so they must be equal. As a consequence, $N_{ab} = N_{cd}$ and $N_{ba} = N_{dc}$, so $(N_{ab}, N_{ba}) \sim_D (N_{cd}, N_{dc})$.

2. $M_{ab} > M_{cd}$ and $N_{ab} + N_{ba} = N_{cd} + N_{dc}$. Then it holds that $M_{ab} = N_{ab} - N_{ba} = (N_{ab} + N_{ba}) - 2N_{ba}$ and for the same reason $N_{cd} - N_{dc} = (N_{cd} + N_{dc}) - 2N_{dc} = (N_{ab} + N_{ba}) - 2N_{dc}$. Thus, $N_{dc} > N_{ba}$ and $N_{cd} < N_{ba}$. Hence, $(N_{ab}, N_{ba}) >_D (N_{cd}, N_{dc})$ by monotonicity.

3. $M_{ab} > M_{cd}$ and $N_{ab} + N_{ba} = |V| = N_{cd} + N_{dc} + 1$. With the same steps as in case 2, we get that $2N_{ba} < 2N_{dc} + 1$. Thus, $N_{ba} \leq N_{dc}$ and $N_{ab} > N_{cd}$, so $(N_{ab}, N_{ba}) >_D (N_{cd}, N_{dc})$ by monotonicity.

4. $M_{ab} > M_{cd}$ and $N_{ab} + N_{ba} + 1 = |V| = N_{cd} + N_{dc}$. With the same steps, we get that $2N_{ba} + 1 < 2N_{dc}$. Thus, $N_{ba} < N_{dc}$ and $N_{ab} \geq N_{cd}$, so $(N_{ab}, N_{ba}) >_D (N_{cd}, N_{dc})$ by monotonicity. ☐

We will begin with the case that all non-manipulators vote complete rankings. Then, the problem's complexity differs if the number of manipulators is one, or if there are more manipulators.

Lemma 4.18. *Let the defeat strength be monotonic.*
UNWEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION *is P-hard under log-space reduction as soon as there are 2 manipulators, even if all non-manipulators are required to vote complete rankings.*

Proof. Let $k \geq 2$ be the number of manipulators. We only need to analyze the behaviour for the margins measure, for all monotonic defeat strength measures are equivalent if all voters vote complete rankings (cf. Proposition 4.17).

We reduce DUAL HORNSAT to the UNWEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION problem. In the DUAL HORNSAT problem, we are given a formula F with dual Horn clauses $C_1, ..., C_m$ and variables $x_1, ..., x_n$. A dual Horn clause is a clause that contains at most one negated literal.

We formulate the following reduction:

- Candidates $C = \{p, x_1, ..., x_n, x'_1, ..., x'_n, C_1, ..., C_m\}$.
- A voting profile V_{N} such that the margins pairwise matrix $M = M(V_{\text{N}})$ has the following values:
  - $M_{px_i} = 20 + 6k$ and $M_{x_i x'_i} = 10 + 4k$ for all variables $x_1, ..., x_n$.

  - $M_{x'_iC_j} = 8 + 4k$ if a positive literal in clause C_j is x_i.
  - $M_{C_jx_i} = 6 + 6k$ if $\overline{x_i}$ is the only negated literal in clause C_j.
  - $M_{C_jp} = 6 + 6k$ if clause C_j does not contain a negated literal.
  - $M_{pC_j} = 10 + 2k$ for all clauses C_j.
  - All other fields are set to zero, as long as it doesn't contradict that M is skew-symmetric.

- k manipulators.

- p is to be made the unique winner by the manipulators.

We can compute such a voting profile V as required by the margins pairwise matrix using McGarvey's trick [McG53]:
Let $c_1, ..., c_\ell$ be the candidates. If we want to increase M_{st} (therefore, decrease M_{ts}) by ΔM_{st} for candidates $s \neq t$ without changing the margins pairwise matrix in any other way, we introduce voters which vote $s > t > c_{i_1} > ... > c_{i_{\ell-2}}$ and $c_{i_1} > ... c_{i_{\ell-2}} > s > t$ alternately $\frac{\Delta M_{st}}{2}$ times, where $\{c_{i_1}, ..., c_{i_{\ell-2}}\} = C \setminus \{s, t\}$. This technique requires that the increase ΔM_{st} is even, however, this is the case in our reduction. Moreover, we do not need to know the concrete values of any other cell of the matrix, but only the increase in one cell, so we can apply this technique to construct a voting profile as required in logarithmic space, for the number of cells and the values in the cells is polynomial in n, m and k.

It remains to be shown that F is satisfiable if and only if p can become the unique winner:

$\Longrightarrow$: Suppose the formula F is satisfiable by assignment $I : \{x_1, ..., x_n\} \to \{0, 1\}$. All manipulators will vote in their voting profile V_{M} the complete ranking R which is subject to the following conditions:

  - p is ranked first in R.
  - $x_i >_R x'_i$ if and only if x_i is true in I, i.e., $I(x_i) = 1$.
  - $x'_i >_R C_j$ if and only if clause C_j contains positive literal x_i and $I(x_i) = 1$.

Since p is ranked first, we know that a link and thus path to p can have at most strength $6 + 5k$, i.e., $S_{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(c, p) \leq 6 + 5k$ for all candidates $c \in C \setminus \{p\}$. Therefore, all candidates $x_1, ..., x_n$ are beaten by p, for $M(V_{\mathrm{N}} \circ V_{\mathrm{M}})_{px_i} = 20 + 7k > 6 + 5k$. Moreover, $x'_1, ..., x'_n$ must have been defeated, too, since

$$\begin{aligned} S_{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(p, x'_i) &= M(V_{\mathrm{N}} \circ V_{\mathrm{M}})_{x_ix'_i} \\ &> \max\{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})_{x'_iC_j} |\text{ clause } C_j\} \\ &\geq S_{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(x'_i, p) \; . \end{aligned}$$

Finally, we have to show that all candidates $C_1, ..., C_m$ are defeated, too. Suppose there is a positive literal x_i in clause C_j with $I(x_i) = 1$. Then,

$$\begin{aligned} S_{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(p, C_j) &\geq s_{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(p, x_i, x_i', C_j) \\ &= 8 + 5k > 6 + 5k \\ &\geq S_{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(C_j, p) \ . \end{aligned}$$

Otherwise, there must be a negated literal $\overline{x_i}$ in clause C_j with $I(x_i) = 0$. Then,

$$S_{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(p, C_j) \geq M(V_{\mathrm{N}} \circ V_{\mathrm{M}})_{pC_j} = 10 + 3k > 8 + 3k \geq S_{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(C_j, p) \ ,$$

for the only link from C_j with positive margin was the link (C_j, x_i), so only a path over x_i could have a strength greater than k. However, the link (x_i, x_i') has at most strength $M(V_{\mathrm{N}} \circ V_{\mathrm{M}}) = 8 + 3k$.

Hence, p beats any candidate, so there is an unweighted constructive unique winner manipulation.

$\Longleftarrow$: Suppose V_{M} is the manipulators' voting profile of an unweighted constructive unique winner manipulation. Without loss of generality, assume that all manipulators voted the same complete ranking R, for all manipulators can vote the same complete ranking by Theorem 4.16. Moreover, the Schulze method satisfies the Monotonicity criterion (cf. [Sch11]), so we also assume that p is ranked first in R. Just as in the other direction, the strength from any candidate to p is then at most $6 + 5k$, i.e., $S_{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(c, p) \leq 6 + 5k$.

We define the assignment $I(x_i) = 1 \iff S_{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(x_i, p) \geq 10 + 3k$ and show that it satisfies the formula F:

Let C_j be some clause. If C_j contains the negated literal $\overline{x_i}$ with $I(x_i) = 0$, C_j is satisfied. If it does not contain a negated literal or the literal is false, then $S_{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(C_j, p) \geq 10 + 3k$. Therefore, there must be a (p, C_j)-path whose strength exceeds $10 + 3k$. The only paths which can have such a strength lead over a candidate x_i, in other words are paths of the form $P = (p, x_i, x_i', C_j)$. Assume $S_{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(P) = S_{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(p, C_j) > S_{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(C_j, p)$. Then, the path (x_i, x_i', C_j) has a strength greater than $10 + 3k$, and since there is a (C_j, p)-path of at least strength $10 + 3k$, there is a (x_i, p)-path of at least strength $10 + 3k$. Thus, $I(x_i) = 1$. Therefore, C_j is satisfied, since there is a positive literal which is true.

Hence, the formula F is satisfied, for all clauses are satisfied. □

Lemma 4.19. *Let the defeat strength be monotonic.*
UNWEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION *is* NL-*complete for one manipulator if all non-manipulators are required to vote complete rankings.*

Proof. For the membership of NL, consider the result of Theorem 4.3. From the theorem follows that we only need to compute the strength of the strongest path from p to any other candidate and from any other candidate to p. Algorithm 5 can compute this strength.

Algorithm 5: Algorithm for Computing Maximum Path Strength

```
Input: Finite, non-empty set of candidates $C$
       Non-manipulators' voting profile $V$
       Candidates $s, t \in C$
Output: The strength of the strongest path from $s$ to $t$, i.e. $S_{M(V)}(s,t)$
begin
    strength ← $\min\{M(V)_{ab} | a, b \in C\}$;
    foreach $a, b \in C; a \neq b$ do
        if strength < $M(V)_{ab}$ then
            if there is an $(s,t)$-path of at least strength $M(V)_{ab}$ then
                strength ← $M(V)_{ab}$;
    return strength;
```

This algorithm needs at most logarithmic space: We do not need to save the whole pairwise matrix, but instead can calculate each cell on demand. Moreover, the variable `strength` needs at most $\mathcal{O}(\log m)$ space, where m is the number of non-manipulators. Finally, testing whether there is an (s,t)-path of at least strength $M(V)_{ab}$ is an NL-complete problem, for this is the same as if we asked whether there is an (s,t)-path in graph $G = (C, E)$ with edge set $E = \{(x,y) | x, y \in C, M(V)_{xy} \geq M(V)_{ab}\}$, and STCON is NL-complete. Thus, this can also be done in logarithmic space. Since NL = coNL [Imm88; Sze88], the algorithm also needs at most logarithmic space, if there is no such path. Thus, the algorithm, and hence the test of Theorem 4.3, needs at most logarithmic space on a non-deterministic Turing machine.

Next, we prove that the problem is NL-hard. We reduce STCON to the UNWEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION with one manipulator and all non-manipulators voting complete rankings. We are given a directed graph $G = (V, E)$ and are asked whether there is an (s,t)-path in G. We construct the following reduction:

- Candidates $C = V \uplus E$.
- A voting profile V_{N} of the non-manipulators such that the margins pairwise

matrix $M = M(V_{\mathrm{M}})$ is

$$M_{ab} = \begin{cases} 2 & a = t \wedge b \in V \setminus \{t\} \\ 2 & a \in V \wedge (\exists c \in V)[(a,c) = b] \\ 2 & b \in V \wedge (\exists c \in V)[(c,b) = a] \\ -2 & \text{if } M_{ba} = 2 \\ 0 & \text{otherwise} \end{cases} .$$

We can use McGarvey's trick again to calculate such a profile of complete rankings in logarithmic space.

- s shall become the unique winner.

The construction is of such a form that $P = (v_0, ..., v_\ell)$ is a simple (s,t)-path in G if and only if $P' = (v_0, (v_0, v_1), v_1, ..., (v_{\ell-1}, v_\ell), v_\ell)$ is an (s,t)-path with strength 2. Moreover, all simple (s,t)-paths in the constructed voting situation follow this rule if they have at least strength 2. Therefore, there is an (s,t)-path in G if and only if there is an (s,t)-path in the voting situation of at least strength 2. Furthermore, if there is an (s,t)-path in the voting situation of at least strength 2, then there is an (s,a)-path of strength 2 via t (because of the first case in the definition of the pairwise matrix). Since there is no path of strength greater than 2 and there is a (t,s)-path of strength 2, we can also say that there is an (s,t)-path in G if and only if $S_M(s,c) \geq S_M(c,s)$ for all candidates $c \in C \setminus \{s\}$. Because different values in the margins pairwise matrix have at least distance 2, it is equivalent to say that there is an (s,t)-path in G if and only if $S_M(s,c) + 1 > S_M(c,s) - 1$ for all candidates $c \in C \setminus \{s\}$. By Theorem 4.3, this is just the criterion for the existence of a successful manipulation, so there is a successful manipulation if and only if there is an (s,t)-path in G. Hence, the problem is NL-hard. □

Next, we consider the case that the voters can also vote any partial ranking. If this is the case, then also the case for one manipulator becomes more difficult:

Lemma 4.20. *Let the defeat strength be monotonic.*
UNWEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION *is P-hard under log-space reduction for any number of manipulators, if the non-manipulators are allowed to vote any partial ranking.*

Proof. For at least 2 manipulators, Lemma 4.18 applies. Therefore, we only have to consider the case if there is exactly one manipulator.

The proof is analogous to the proof of Lemma 4.18. We only have to adjust the voting profile V_{N}: We do now require for its margins pairwise matrix $M = M(V_{\mathrm{N}})$ that

- $M_{px_i} = 10$ and $M_{x_i x'_i} = 6$ for all variables $x_1, ..., x_n$.
- $M_{x'_i C_j} = 5$ if the positive literal x_i is in clause C_j.
- $M_{C_j x_i} = 6$ if $\overline{x_i}$ is the only negated literal in C_j.
- $M_{C_j p} = 6$ if clause C_j does not contain a negated literal
- $M_{pC_j} = 4$ for all clauses C_j.
- All other fields are set to zero, as long as it doesn't contradict that M is skew-symmetric.

We slightly change the way we compute the voting profile: Let the candidates be $c_1, ..., c_\ell$. We first construct the voting profile the same way as in Lemma 4.18's proof, until we increased the link's margin to the nearest even number not exceeding the requested margin ΔM_{st}. If ΔM_{st} is odd, then we also add two votes $s > t > c_{i_1} > ... > c_{i_{\ell-2}}$ and $c_{i_1} > ... c_{i_{\ell-2}} > s \sim t$. This vote increases M_{st} by one, but doesn't change all other margins. The voting profile can also be constructed in logarithmic space, as the analysis in the original proof still applies.

By Theorem 4.16, the manipulator can vote a complete ranking. Thus, it suffices to analyze the reduction with the margins measure because of Proposition 4.17.

Later on in the proof, we derive the assignment I from a successful manipulation by defining $I(x_i) = 1 \iff S_{M(V_M \circ V_N)}(x_i, p) \geq 5$. □

If we combine all results, we get a full characterization of the difficulty of the UNWEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION problem, depending on the number of manipulators and what kind of votes the voters can vote.

Theorem 4.21. *Let the defeat strength be monotonic.*

1. *If all non-manipulators are required to vote complete rankings,* UNWEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION *is NL-complete, if there is* 1 *manipulator, and P-complete, if there are* 2 *or more manipulators.*
2. UNWEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION *is P-complete, if non-manipulators can vote any partial ranking.*

Proof.

1. If there is one manipulator, the problem is NL-complete due to Lemma 4.19. If there are at least two manipulators, Lemma 4.18 tells us that the problem is P-hard, and since there is a polynomial time algorithm for the problem, UNWEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION is P-complete.

2. By Lemma 4.20, the problem is P-hard for any number of manipulators, and since here is a polynomial time algorithm, UNWEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION is P-complete. □

4.2 Weighted Voting

Manipulators may not always find themselves in an environment where votes are unweighted. Sometimes, some votes have a greater weight than others. This can make the problem to manipulate the election much harder.

In the special case of the Schulze method, however, the problem's complexity remains unchanged, even if the votes become weighted.

Theorem 4.22. *Let* $I = (V_{\mathrm{N}}, W_{\mathrm{N}}, M, W_{\mathrm{M}}, p)$ *be a weighted manipulation instance with* $k = \|M\|$ *manipulators. Let the defeat strength be monotonic.*

Let M' *be a set of manipulators with* $\|M'\| = \sum_{i=1}^{k} (W_{\mathrm{M}})_i$ *manipulators. There is a weighted constructive unique winner manipulation for* I *if and only if there is an unweighted constructive unique winner manipulation for unweighted manipulation instance* $I' = (E(V_{\mathrm{N}}, W_{\mathrm{N}}), M', p)$.

Proof.

$\Longrightarrow$: Suppose there is a weighted constructive unique winner manipulation (I, V_{M}). Since the winner is calculated by applying the Schulze method to the expansion of the weighted voting profile, it follows that p is the unique winner for unweighted voting profile $E((V_{\mathrm{N}}, W_{\mathrm{N}}) \circ (V_{\mathrm{M}}, W_{\mathrm{M}}))$. Then,

$$E((V_{\mathrm{N}}, W_{\mathrm{N}}) \circ (V_{\mathrm{M}}, W_{\mathrm{M}})) = E(V_{\mathrm{N}}, W_{\mathrm{N}}) \circ E(V_{\mathrm{M}}, W_{\mathrm{M}}) \ . \tag{4.18}$$

Since $E(V_{\mathrm{N}}, W_{\mathrm{N}})$ is the non-manipulators' voting profile in I' and $E(V_{\mathrm{M}}, W_{\mathrm{M}})$ is an unweighted $\|M'\|$-voter voting profile and p is the unique winner if both voting profiles are concatenated, there is an unweighted constructive unique winner manipulation for I'.

$\Longleftarrow$: Suppose there is an unweighted constructive unique winner manipulation (I', V_{M}). By Theorem 4.16, there is also one where all manipulators vote the same complete ranking R, so w.l.o.g. assume that $V_{\mathrm{M}} = (R, ..., R)$. Consider the weighted profile $(V'_{\mathrm{M}}, W_{\mathrm{M}})$ with $V'_{\mathrm{M}} = (R, ..., R) \in \mathcal{T}^k$. Then, $E(V'_{\mathrm{M}}, W_{\mathrm{M}}) = V_{\mathrm{M}}$. Consequently, the expansion of the weighted profile for instance I takes the form

$$E((V_{\mathrm{N}}, W_{\mathrm{N}}) \circ (V'_{\mathrm{M}}, W_{\mathrm{M}})) = E(V_{\mathrm{N}}, W_{\mathrm{N}}) \circ V_{\mathrm{M}} \ . \tag{4.19}$$

Thus, the Schulze method is applied to the same voting profile in the weighted and the unweighted case. Since p is a unique winner for this profile, (I, V'_{M}) is a weighted constructive unique winner manipulation. ☐

Therefore, we can decide the WEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION problem by deciding the UNWEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION problem. Furthermore, as seen in the reasoning in the proof of Theorem 4.22, we can also use the algorithms to find an unweighted manipulation for finding a weighted manipulation: We only have to change the number of manipulators to $\sum_{i=1}^{k}(W_{\mathrm{M}})_i$ and they can be used to produce a weighted manipulation.

Theorem 4.23. *Let* $(V_{\mathrm{N}}, W_{\mathrm{N}}, M, W_{\mathrm{M}}, p)$ *be a weighted manipulation instance with* m *non-manipulators and* $k = \|M\|$ *manipulators. Let the defeat strength be monotonic.*

1. *The* WEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION *problem is P-complete.*

2. *A weighted constructive unique winner manipulation can be computed in* $\mathcal{O}(n^2(n+m+k))$ *time and* $\mathcal{O}(kn^2)$ *space, if one exists.*

Proof. First, the hardness results for the unweighted problem carry over to the weighted manipulation problem. Since one manipulator can have a vote weight greater than 1, the problem is at least P-hard.

To find a manipulation, we can reuse the algorithms for the unweighted case for the weighted case, too. If we use Algorithm 4 to compute an unweighted manipulation, we can easily also use it for an unweighted manipulation if we use the same transformation as in Theorem 4.22, for the algorithm always produces a ranking which can be voted by all manipulators. Additionally, we only need to compute the Condorcet pairwise matrix and the voting profile for the manipulators, which take $\mathcal{O}(mn^2)$ and $\mathcal{O}(kn^2)$ time and $\mathcal{O}(n^2)$ and $\mathcal{O}(kn^2)$ space respectively. ☐

5 Constructive Co-Winner Manipulation

It isn't always possible that a candidate can be made the only winner by some manipulators. In this case, they have to make do with making the only a winner of potentially many other winners. We will examine this weaker form of constructive manipulation and find solution to it in this chapter. As we will find out, this problem can be solved if we adapt the approach to the CONSTRUCTIVE UNIQUE WINNER MANIPULATION problem to this problem.

5.1 Unweighted Voting

As in the previous chapter, we will begin with the case that votes are unweighted.

We can express the meaning of an unweighted constructive co-winner manipulation directly for the Schulze method:

Proposition 5.1. *Let* $I = (V_{\mathrm{N}}, M, p)$ *an unweighted manipulation instance with* $k = \|M\|$ *manipulators.*

A pair (I, V_{M}) *is an unweighted constructive co-winner manipulation if and only if*

$$S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(p, c) \gtrsim_D S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(c, p) \text{ for all } c \in C \setminus \{p\} \ . \tag{5.1}$$

Proof. This is a direct consequence of Proposition 3.4. □

This proposition gives us a tool to check whether some manipulators' voting profile successfully manipulates the election. However, it does not give us an efficient way to find out if any such profile exists.

To determine if we can efficiently find a manipulation, we need a different approach.

5.1.1 Weak Manipulation Thresholds

As already shortly mentioned, we will adapt the approach to the UNWEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION problem to solve this problem. There, we introduced the concept of a strong manipulation threshold and developed rules so that a manipulation threshold could be further refined.

In a co-winner manipulation the strength of the strongest path from p to any other candidate c does only need to match the strength of the strongest path from p to candidate c. A simple consequence is that the thresholds for this problem only need to be at least as great as the strength of the strongest path from p to any candidate c in all manipulations, but they need not exceed the strength.

Definition 5.2. Let (V_{N}, M, p) an unweighted manipulation instance with $k = \|M\|$ manipulators.

1. A *weak manipulation threshold* is a function $\theta : C \setminus \{p\} \to \mathbb{N} \times \mathbb{N}$ such that if there is an unweighted constructive co-winner manipulation, then the value $\theta(c)$ is at least as great as the strength of any (c, p)-path in an unweighted constructive unique winner manipulation, i.e., for all such manipulations (I, V_{M}) it holds that
$$\theta(c) \gtrsim_D S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(c, p) \text{ for all } c \in C \setminus \{p\} . \tag{5.2}$$

2. A weak manipulation threshold θ is *tight* if there is an unweighted constructive co-winner manipulation if and only if
$$\theta(c) \gtrsim_D S_{D(V_{\mathrm{N}})+kI^-}(c, p) \text{ for all } c \in C \setminus \{p\} . \tag{5.3}$$

If we know a weak manipulation threshold, we can formulate a necessary condition for the existence of an unweighted constructive co-winner manipulation:

Proposition 5.3. *Let $I = (V_{\mathrm{N}}, M, p)$ be a manipulation instance with $k = \|M\|$ manipulators. Let θ be a weak manipulation threshold. Let $>_D$ be monotonic.*

If there is an unweighted constructive co-winner manipulation, then
$$\theta(c) \gtrsim_D S_{D(V_{\mathrm{N}})+kI^-}(c, p) \textit{ for all } c \in C \setminus \{p\} .$$

Proof. Suppose there is a manipulation (I, V_{M}). Then,
$$\theta(c) \gtrsim_D S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(c, p) \gtrsim_D S_{D(V_{\mathrm{N}})+kI^-}(c, p) \text{ for all } c \in C \setminus \{p\} . \qquad \square$$

Example 5.1.

1. $S_{D+kI^+}(p,\cdot)$ is a weak manipulation threshold, as we know by Proposition 5.1 that for any manipulation (I, V_{M})

$$S_{D+kI^+}(p,c) \gtrsim_D S_{D(V_{\mathrm{N}}\circ V_{\mathrm{M}})}(p,c) \gtrsim_D S_{D(V_{\mathrm{N}}\circ V_{\mathrm{M}})}(c,p) \text{ for all } c \in C \setminus \{p\} .$$

2. $\theta : c \mapsto \max_D\{(D+kI^+)_{ab}|a,b \in C\}$ is also a weak manipulation threshold, because $S_{D+kI^+}(p,c) \lesssim_D \max_D\{(D+kI^+)_{ab}|a,b \in C\}$ for all candidates $c \in C \setminus \{p\}$.

3. A tight threshold is

$$\theta : c \mapsto \begin{cases} S_{D+kI^+}(p,c) & \text{there is a manipulation} \\ S_{D+(k+1)I^-}(c,p) & \text{otherwise} \end{cases} .$$

However, we do not know how to efficiently compute this threshold.

Aim. We are looking for an efficiently computable weak manipulation threshold that is tight.

5.1.2 The Strongly Restricting Link Rule

The efficiently computable thresholds up to here are sadly not tight, but we are aiming for finding a tight one. To find a tight threshold, we either have to find a completely new threshold or we are gradually refining the currently known weak manipulation thresholds until they have become tight.

For strong manipulation thresholds, we already found two rules that can refine such a threshold: The Restricting Link rule and the Incoming Paths rule. We can adopt the Incoming Paths rule for weak manipulation thresholds, for this rule also works for weak manipulation thresholds, although the rationale, why the rule is correct, has to be slightly adapted. The Restricting Link rule in its original form, however, falls apart.

Nevertheless, we can find a variant of the Restricting Link rule that works on weak manipulation thresholds. Again, assume that we have two candidates $s, t \in C \setminus \{p\}$ with $\theta(s) <_D \theta(t)$. For some reason, the strength of the path from t to some other candidate has to be increased in a manipulation; this results in an increase of the strength of the strongest path from t to p. If we are unlucky, this strength is fed back to candidate s over the link (s,t). This feedback can only be mitigated by reducing the strength of the link (s,t) until the link (s,t) has at most strength $\theta(s)$. This is the difference to the Restricting Link rule: The problematic feedback for co-winner manipulations cannot arise if (s,t) has the same strength as

$\theta(s)$, whilst the negative feedback still happens for unique winner manipulations if (s, t) has the same strength as $\theta(s)$. We will call links like (s, t) *strongly restricting links*.

Definition 5.4. Let (V, M, p) be an unweighted manipulation instance with $k = \|M\|$ manipulators. Let $D = D(V)$.

A link (a, b) is *strongly ρ-restricting* if and only if $(D + kI^-)_{ab} >_D \rho$.

Remark 5.2. In particular, if the defeat strength measure of choice is the margins measure, a strongly ρ-restricting link (a, b) has at least strength $s_M(P) - k > \rho$. Equivalently, a strongly ρ-restricting link can also be seen as $(\rho + 1)$-restricting, if the margins measure is used.

Strongly Restricting Link rule. Let $s \in C \setminus \{p\}$ be a source candidate and $t \in C \setminus \{p\}$ be a target candidate. If $\theta(s) <_D \theta(t)$ and the link (s, t) is strongly $\theta(s)$-restricting,, then assign the new threshold value $\theta(s)$ to the candidate t.

Proposition 5.5. *Let $I = (V_{\mathrm{N}}, M, p)$ be an unweighted manipulation instance. Let the defeat strength be monotonic.*

Given a weak manipulation threshold θ, an application of the Strongly Restricting Link rule produces a weak manipulation threshold.

Proof. The proof is analogous to the proof of Proposition 4.7 that shows the correctness of the Restricting Link rule for manipulation thresholds. ☐

5.1.3 The Match Threshold

We acquire a new, tighter weak manipulation threshold if we apply any of the rules to a known weak manipulation threshold. If we start from one well-known threshold and apply both rules as often as possible, we gain a new threshold which is as small as it can get using the initial threshold and the rules.

Algorithm 6: General Algorithm for Match Threshold Computation

```
Input: Finite, non-empty set of candidates $C$
       Defeat strength pairwise matrix $D \in (\mathbb{N} \times \mathbb{N})^{C \times C}$
       Number of manipulators $k$
       A candidate $p \in C$ who is preferred by the manipulators
Output: The match threshold $\mathrm{mt} : C \setminus \{p\} \to \mathbb{N} \times \mathbb{N}$
begin
    $\mathrm{mt} \equiv \max_D\{(D + kI^+)_{ab} | a, b \in C\}$;
    while the Strongly Restricting Link rule or the Incoming Paths rule is applicable to mt do
        apply the rule;
    return mt;
```

Proposition 5.6. *Algorithm 6 terminates.*

Proof. The proof is analogous to Proposition 4.9's proof. □

The algorithm yields the *match threshold* mt. The match threshold is a weak manipulation threshold, so we can use it to test whether there is no manipulation.

Lemma 5.7. *Let $I = (V_{\mathrm{N}}, M, p)$ be an unweighted manipulation instance. Let the defeat strength be monotonic.*

The match threshold is a weak manipulation threshold.

Proof. The proof is analogous to Lemma 4.10's proof. □

Algorithm 6 leaves the order open in which the rules should be applied. If we follow the same strategy and steps as we already did to derive Algorithm 3 for the beat threshold, we derive Algorithm 7, an algorithm that efficiently computes the match threshold.

Lemma 5.8. *Algorithm 7 computes the match threshold in $\mathcal{O}(n^3)$ time and $\mathcal{O}(n)$ space.*

Proof. The correctness and runtime proofs are analogous to the proofs for Algorithm 3 that efficiently computes the beat threshold. In the proofs, we simply have to replace the Restricting Link rule by the Strongly Restricting Link rule. □

5.1.4 Computing a Manipulation

Like the beat threshold for the constructive unique winner manipulation, we can also use the match threshold to find a constructive co-winner manipulation. Lemma 5.9 provides that there is a manipulation we searched for, given some precondition.

Lemma 5.9. *Let $I = (V_{\mathrm{N}}, M, p)$ be an unweighted manipulation instance with $k = \|M\|$ manipulators. Let $D = D(V_{\mathrm{N}})$.*

There is an unweighted constructive co-winner manipulation, if

$$(D + kI^{-})_{cp} \lesssim_D \mathrm{mt}(c) \text{ for all } c \in C \setminus \{p\} . \tag{5.4}$$

Proof. The proof is analogous to the proof of Lemma 4.14, mainly exchanging the match threshold for the beat threshold. The only part with substantial changes is when we determine maximal strength of any (c, p)-path for any candidate $c \in C \setminus \{p\}$. In that case, we can only show that $S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(c, p) \lesssim_D \mathrm{mt}(c)$ for all candidates $c \in C \setminus \{p\}$, for the Strongly Restricting Link rule is weaker than the Restricting Link rule, and the precondition (5.4) is weaker than the precondition of Lemma 4.14. The resulting changes follow naturally. □

```
Algorithm 7: Algorithm for Efficient Computation of the Match Threshold
Input: Finite, non-empty set of candidates $C$
       Defeat strength pairwise matrix $D \in (\mathbb{N} \times \mathbb{N})^{C \times C}$
       Number of manipulators $k$
       A candidate $p \in C$ who is preferred by the manipulators
Data: Priority queue $Q$
Output: Match threshold $\mathrm{mt} : C \setminus \{p\} \to \mathbb{N} \times \mathbb{N}$
begin
    $\mathrm{mt} \equiv \max_D\{(D + kI^+)_{ab} | a, b \in C\}$;
    repeat
        $\mathrm{mt}'(\cdot) \leftarrow$ compute path strength from $p$ in weighted digraph
        $G = (C, E; \lambda)$ with edge set $E = C \times (C \setminus \{p\})$ and weights
        $\lambda : E \to \mathbb{N} \times \mathbb{N} : (a, b) \mapsto \min_D\{(D + kI^+)_{ab}, \mathrm{mt}(b)\}$;
        foreach $c \in C \setminus \{p\}$ do
            $Q$.insert($c$, $\mathrm{mt}'(c)$);
        stable $\leftarrow$ true;
        while $Q \neq \emptyset$ do
            $c \leftarrow Q$.extractMin();
            foreach $a \in C \setminus \{c\}$ do
                if $\mathrm{mt}'(a) >_D \mathrm{mt}'(c)$ and $\mathrm{mt}'(c) <_D (D + kI^-)_{ca}$ then
                    $\mathrm{mt}'(a) \leftarrow \mathrm{mt}'(c)$;
                    $Q$.updateKey($a$, $\mathrm{mt}'(a)$);
                    stable $\leftarrow$ false;
        $\mathrm{mt} \leftarrow \mathrm{mt}'$;
    until stable;
    return mt;
```

This sufficient condition leads us to the central structural result for the existence of an unweighted constructive co-winner manipulation:

Theorem 5.10. *Let (V, M, p) be an unweighted manipulation instance with $k = \|M\|$ manipulators. Let $D = D(V)$. Let the defeat strength be monotonic.*

1. *The match threshold is a weak manipulation threshold that is tight.*
2. *There is an unweighted constructive co-winner manipulation if and only if*

$$(D + kI^-)_{cp} \lesssim_D \mathrm{mt}(c) \text{ for all } c \in C \setminus \{p\} . \tag{5.5}$$

Proof. Both statements use that the path over the edge connecting c and p is at most as strong as the strongest path, so $(D + kI^-)_{cp} \lesssim_D S_{D+kI^-}(c,p)$. From this and since the match threshold is a weak manipulation threshold, it also follows that the manipulation threshold is tight by Lemma 5.9. The second statement follows from Lemma 5.9 and Proposition 5.3. □

As we did for the unweighted constructive unique winner manipulation, we can use the same construction also to compute an unweighted constructive co-winner manipulation, if one exists. This yields Algorithm 8.

Algorithm 8: Algorithm for Unweighted Constructive Co-Winner Manipulation of the Schulze method

```
Input: Finite, non-empty set of candidates $C$
       Defeat strength pairwise matrix $D \in (\mathbb{N} \times \mathbb{N})^{C \times C}$
       Number of manipulators $k$
       A candidate $p \in C$ who is preferred by the manipulators
Output: A complete ranking $R$ such that $(R, ..., R) \in \mathcal{T}^k$ is the manipulators'
        voting profile for a constructive co-winner manipulation, or nothing
        if there is no manipulation
begin
    $\mathrm{mt}(\cdot) \leftarrow$ compute match threshold;
    if there ex. $c \in C \setminus \{p\}$ with $(D_{cp} + kI^-) >_D \mathrm{mt}(c)$ then
        return no unweighted constructive unique winner manipulation;
    Let $G = (C, E)$ be a directed graph with
        $E := \{(p,a) | a \in C \setminus \{p\} \wedge (D + kI^+)_{pa} \gtrsim_D \mathrm{mt}(a)\} \cup$
             $\{(a,b) | a,b \in C \setminus \{p\} \wedge \min_D\{\mathrm{mt}(a), (D + kI^+)_{ab}\} \gtrsim_D \mathrm{mt}(b)\}$ ;
    Let $T \subseteq G$ be a spanning arborescence rooted at $p$;
    Let $R$ be topological ordering of $T$ such that $\mathrm{mt}(a) >_D \mathrm{mt}(b) \implies a >_R b$
        for all $a, b \in C \setminus \{p\}$;
    return $R$;
```

Theorem 5.11. *Let the defeat strength be monotonic. Let (V, M, p) be a manipulation instance with $k = \|M\|$ manipulators and m non-manipulators.*

1. *Algorithm 8 computes an unweighted constructive co-winner manipulation's ballot for all manipulators in $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space, if one exists.*
2. *An unweighted constructive co-winner manipulation can be computed in $\mathcal{O}(n^2(n + m + k))$ time and $\mathcal{O}(kn^2)$ space.*
3. *If there is an unweighted constructive unique winner manipulation, then all manipulators can vote the same complete ranking to successfully manipulate.*

Proof.

1. For the correctness, cf. Lemma 5.9's proof and Theorem 5.10. The match threshold can be computed in $\mathcal{O}(n^3)$ time (by Lemma 5.8) and $\mathcal{O}(n)$ space. The graph G, its arborescence T and the topological ordering R can also be computed in $\mathcal{O}(n^2)$ time using $\mathcal{O}(n^2)$ space. Therefore, Algorithm 8 runs in $\mathcal{O}(n^3)$ time and $\mathcal{O}(n^2)$ space.
2. Additionally, we need to compute the Condorcet pairwise matrix, which takes $\mathcal{O}(mn^2)$ time and $\mathcal{O}(n^2)$ space. Moreover, we need to construct the manipulators' voting profile. Depending on the way the profile is represented, we have to make a copy of the ranking for each manipulator, which can take at most $\mathcal{O}(kn^2)$ time and needs at most $\mathcal{O}(kn^2)$ space. Hence, we need at most $\mathcal{O}(n^2(n + m + k))$ time.
3. Algorithm 8 always computes a complete ranking that should be voted by all manipulators. □

5.1.5 Complexity

Whilst we found a polynomial time algorithm, this did not give us the complexity class for the UNWEIGHTED CONSTRUCTIVE CO-WINNER MANIPULATION problem, except that it is in P. As we will see, this problem is one of the hardest problems that can be solved in polynomial time.

Theorem 5.12. *Let the defeat strength be monotonic.*

UNWEIGHTED CONSTRUCTIVE CO-WINNER MANIPULATION *is P-complete for any number of manipulators, even if all voters have to vote complete rankings.*

Proof. The problem is in P, for there is a polynomial time algorithm (see Theorem 5.11).

To show that the problem is P-hard, we adapt Lemma 4.18's proof.

In the reduction, we use a voting profile V_{N} such that its margins pairwise matrix $M = M(V_{\mathrm{N}})$ takes the form:

- $M_{px_i} = 20 + 6k$ and $M_{x_i x_i'} = 10 + 4k$ for all variables $x_1, ..., x_n$.
- $M_{x_i' C_j} = 10 + 4k$ if the non-negated literal x_i is in clause C_j.
- $M_{C_j x_i} = 10 + 6k$ if $\overline{x_i}$ is the only negated literal in C_j.
- $M_{C_j p} = 10 + 6k$ if clause C_j does not contain a negated literal
- $M_{pC_j} = 10 + 2k$ for all clauses C_j.
- All other fields are set to zero, as long as it doesn't contradict M's skew-symmetry.

Using the same technique as in the original proof, we can construct such a profile that consists of complete rankings in polynomial time.

The remaining proof can be adapted straightforwardly. The assignment I derived from the successful manipulation is $I(x_i) = 1 \iff S_{M(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(x_i, p) > 10 + 3k$. $\square$

5.2 Weighted Voting

In difference to before, the voters now have differing weights. However, the results we found before for the unweighted case can be carried over to the weighted case. Especially helpful is that we determined that all manipulators can vote identically.

Theorem 5.13. *Let* $I = (V_{\mathrm{N}}, W_{\mathrm{N}}, M, W_{\mathrm{M}}, p)$ *be a weighted manipulation instance with* $k = \|M\|$ *manipulators. Let the defeat strength be monotonic.*

Let M' *be a set of manipulators of size* $\|M'\| = \sum_{i=1}^{k}(W_{\mathrm{M}})_i$. *There is a weighted constructive co-winner manipulation for* I *if and only if there is an unweighted constructive co-winner manipulation for unweighted manipulation instance* $I' = (E(V_{\mathrm{N}}, W_{\mathrm{N}}), M', p)$.

Proof. The proof is analogous to the proof of Theorem 4.22: It was shown in Theorem 5.11 that all manipulators can vote identically. ☐

Consequently, the WEIGHTED CONSTRUCTIVE CO-WINNER MANIPULATION and the UNWEIGHTED CONSTRUCTIVE CO-WINNER MANIPULATION decision problems are equally hard. Moreover, we can also use the algorithms to find an unweighted constructive co-winner manipulation to compute a weighted destructive manipulation, based on the construction used in the proof of Theorem 5.13: We simply assume that there are $\sum_{i=1}^{k}(W_{\mathrm{M}})_i$ manipulators.

Theorem 5.14. *Let* $I = (V_{\mathrm{N}}, W_{\mathrm{N}}, M, W_{\mathrm{M}}, p)$ *be a weighted manipulation instance with* m *non-manipulators and* $k = \|M\|$ *manipulators. let the defeat strength be monotonic.*

1. *The* WEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION *problem is P-complete.*
2. *A weighted constructive co-winner manipulation can be computed in* $\mathcal{O}(n^2(n+m+k))$ *time and* $\mathcal{O}(kn^2)$ *space, if one exists.*

Proof. First, hardness results for the unweighted case carry over to the weighted case, for the unweighted manipulation problem is a special case of the weighted manipulation problem. Therefore, the problem is P-hard.

The algorithms for the unweighted case can be used for the weighted problem, too. The Condorcet pairwise matrix for $(V_{\mathrm{N}}, W_{\mathrm{N}})$ can be computed in $\mathcal{O}(mn^2)$ time. The total weight of the manipulators' votes can be computed in $\mathcal{O}(k)$ time. A complete ranking which can be voted by all manipulators can be computed in $\mathcal{O}(n^3)$ time using Algorithm 8, using the total weight of the manipulators as the argument for the number of manipulators. Finally, the manipulators' voting profile has to produced, which takes at most $\mathcal{O}(kn^2)$ time and space. ☐

6 Destructive Manipulation

The manipulators do not always aim to make one candidate a winner; sometimes, they try to prevent a specific candidate from winning. This is destructive manipulation.

For the Schulze method, destructive manipulation has already been studied by Parkes and Xia in [PX12]. They understood destructive manipulation in the sense that the manipulators want to prevent a candidate from becoming the unique winner of the election. Here, we look at a similar, related problem: Can the manipulators prevent a candidate from winning and if so, can they find a manipulation in polynomial time?

Here, it will be explored how and when a candidate can be prevented from winning. We present an alternative version of Parkes' and Xia's proof. This alternative proof, while not fundamentally different, is not identical to the Parkes' and Xia's proof, for it constructs a slightly different voting profile for the manipulators.

6.1 Unweighted Voting

In this section, we will first look for a way to compute a destructive manipulation. Afterwards, we will identify the problem's complexity class.

6.1.1 Computing a Manipulation

As was already the case for constructive manipulation, destructive manipulation has been defined in Sect. 2.3 for all voting rules in general. Just like the other manipulation problems we considered, we can also express directly the notion of a destructive manipulation for the Schulze method.

Proposition 6.1. *Let $I = (V_\mathrm{N}, M, p)$ an unweighted manipulation instance with $k = \|M\|$ manipulators.*

A pair (I, V_M) is an unweighted destructive manipulation if and only if

$$S_{D(V_\mathrm{N} \circ V_\mathrm{M})}(p, c) <_D S_{D(V_\mathrm{N} \circ V_\mathrm{M})}(c, p) \text{ for some } c \in C \setminus \{p\} \ . \tag{6.1}$$

Proof. This is a direct consequence of Proposition 3.4. □

For destructive manipulation to be possible, there must be a candidate c which can beat the disliked candidate p at least if the strength of all paths to p are maximal and the strength of all paths from p is minimal. Obviously, this is a necessary condition, but interestingly, it is also sufficient.

Theorem 6.2. *Let (V_{N}, M, p) an unweighted manipulation instance with $k = \|M\|$ manipulators. Let $D = D(V_{\mathrm{N}})$. Let the defeat strength be monotonic.*

There is an unweighted destructive manipulation if and only if

$$S_{D+kI^+}(c,p) >_D S_{D+kI^-}(p,c) \text{ for some } c \in C \setminus \{p\} \ . \tag{6.2}$$

Proof.

$\Longrightarrow$: Suppose there is an unweighted destructive manipulation with manipulators' voting profile V_{M}. By Proposition 6.1, there is a candidate $c \in C \setminus \{p\}$ such that

$$S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(c,p) >_D S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(p,c) \ .$$

Since $>_D$ is monotonic, we can conclude that

$$S_{D+kI^+}(c,p) \gtrsim_D S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(c,p) >_D S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(p,c) \gtrsim_D S_{D+kI^-}(c,p) \ .$$

Hence, (6.2) holds.

$\Longleftarrow$: Suppose there is a candidate $c \in C \setminus \{p\}$ such that (6.2) holds. There is a strongest simple (c,p)-path $P' = (x_0, ..., x_q)$ for defeat strength pairwise matrix $D + kI^+$ such that all of its subpaths are also strongest paths. There is a last candidate x_i in the path for which (6.2) is true. Then, $(x_i, ..., x_q)$ is a strongest simple (x_i, p)-path with

$$S_{D+kI^-}(p,c_j) \gtrsim_D S_{D+kI^+}(x_j,p) \text{ for all } j \in \{i+1, ..., q-1\} \ .$$

Without loss of generality, let c be a candidate such that there is a strongest simple (c,p)-path P with

$$S_{D+kI^+}(c,p) >_D S_{D+kI^-}(p,c) \text{ and} \tag{6.3}$$

$$S_{D+kI^+}(c_i,p) \lesssim_D S_{D+kI^-}(p,c_i) \text{ for all } i \in \{1, ..., \ell-1\} \ . \tag{6.4}$$

We know that such a c and P exists, for we constructed one in the previous paragraph.

We consider the set of all candidates from which one can reach c with a path that is at least as strong as the strongest possible path from c to p. We name this set Q:

$$Q := \{c\} \cup \{x | \text{ there is a } S_{D+kI^+}(c,p)\text{-restricting } (x,c)\text{-path } \} \ . \tag{6.5}$$

Any candidate $x \in Q$ can reach c via a restricting path, so we know that the strength of some path from x to c must be at least as great as the strongest possible path from c to p, no matter what the manipulators voted:

$$S_{D+kI^+}(c,p) \lesssim_D S_{D+kI^-}(x,c) \; . \tag{6.6}$$

Therefore, by (6.3), $p \notin Q$.

Next, we consider any intermediate vertex c_i of path P (this implies that $i \in \{1,\ldots,\ell-1\}$). From P being a strongest (c,p)-path and (6.4) follows that

$$S_{D+kI^-}(p,c_i) \gtrsim_D S_{D+kI^+}(c_i,p) \gtrsim_D s_{D+kI^+}(c_i,\ldots,c_\ell) \gtrsim_D S_{D+kI^+}(c,p) \; . \tag{6.7}$$

Thus, by (6.6) and (6.7), c_i cannot be an element in set Q, for if c_i were in Q, then

$$S_{D+kI^+}(c,p) \lesssim_D \min_D\{S_{D+kI^-}(p,c_i), S_{D+kI^-}(c_i,c)\} \lesssim_D S_{D+kI^-}(p,c) \; ,$$

which contradicts (6.3).

Let R be a complete ranking such that

$$a \in Q, b \in C \setminus Q \implies a >_R b \text{ and} \tag{6.8}$$

$$i,j \in \{0,\ldots,\ell\}, i < j \implies c_i >_R c_j \; . \tag{6.9}$$

As shown in the previous paragraphs, Q contains no candidate traversed by the path other than the first candidate c, so the complete ranking is well-defined. Let $V_{\mathrm{M}} = (R,\ldots,R) \in \mathcal{T}^k$ be the manipulators' voting profile where all of them vote the ranking R.

Since (6.9) reflects the path P in R's construction, the path P reaches its maximal possible strength, so the path strength from c to p for the manipulators' voting profile is equal to the maximum path strength possible:

$$S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(c,p) \gtrsim_D s_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(P) = s_{D+kI^+}(P) \sim_D S_{D+kI^+}(c,p) \; . \tag{6.10}$$

Let $P' = (x_0,\ldots,x_q)$ be a (p,c)-path. Since $p \notin Q$, but $c \in Q$, there is a last candidate c_i (i.e., $i \in \{0,\ldots,q-1\}$ is maximal) such that c_i is in Q, but c_{i+1} is not. It follows from (6.8) that $c_i <_R c_{i+1}$.

Since $c_i \notin Q$, there is no $S_{D+kI^+}(c,p)$-restricting (c_i,c)-path. Since $c_{i+1} \in Q$, there is such a path or $c_{i+1} = c$. There cannot be a $S_{D+kI^+}(c,p)$-restricting (c_i,c_{i+1})-path, otherwise there would be a $S_{D+kI^+}(c,p)$-restricting (c_i,c)-path and c_i would be an element in the set Q. Therefore,

$$s_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(P') \lesssim_D (D(V_{\mathrm{N}} \circ V_{\mathrm{M}}))_{c_i c_{i+1}} = (D+kI^-)_{c_i c_{i+1}} <_D S_{D+kI^+}(c,p) \; . \tag{6.11}$$

Because (6.11) holds for all (p, c)-paths, we can conclude that

$$S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(c, p) <_D S_{D+kI^+}(p, c) \; . \tag{6.12}$$

From (6.10) and (6.12) follows that

$$S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(p, c) \gtrsim_D S_{D+kI^+}(p, c) >_D S_{D(V_{\mathrm{N}} \circ V_{\mathrm{M}})}(c, p) \; . \tag{6.13}$$

Hence, V_{M} is a manipulators' voting profile of an unweighted destructive manipulation for the given manipulation instance, so there is a manipulation.

□

The proof of Theorem 6.2 constructs a manipulation under the given condition. This construction can also be turned into an algorithm that finds an unweighted destructive manipulation, if one exists.

Algorithm 9: Algorithm for Destructive Manipulation of the Schulze method

```
Input: Finite, non-empty set of candidates $C$
       Defeat strength pairwise matrix $D \in (\mathbb{N} \times \mathbb{N})^{C \times C}$
       Number of manipulators $k$
       A candidate $p \in C$ who is disliked by the manipulators
Output: A complete ranking $R$ s.t. $(R, ..., R) \in \mathcal{T}^k$ is the voting profile of an
        unweighted destructive manipulation, or nothing if there is no
        manipulation
begin
    $S_{D+kI^-}(p, \cdot) \leftarrow$ compute minimum strength from candidate $p$;
    $S_{D+kI^+}(\cdot, p) \leftarrow$ compute maximum strength to candidate $p$;
    if there ex. strongest $(c, p)$-path $P = (c_0, ..., c_\ell)$ s.t.
    $S_{D+kI^+}(c, p) >_D S_{D+kI^-}(p, c)$ and
    $S_{D+kI^+}(c_i, p) \lesssim_D S_{D+kI^-}(p, c_i)$ for $i \in \{1, ..., \ell - 1\}$ then
        $Q \leftarrow \{c\} \cup \{x |$ there ex. $S_{D+kI^+}(c, p)$-restricting $(x, c)$-path $\}$;
        Let $R$ be a complete ranking s.t.

            $a \in Q, b \in C \setminus Q \implies a >_R b$ and
            $i, j \in \{0, ..., \ell\}, i < j \implies c_i >_R c_j$ ;

        return $R$;
    else
        return no unweighted destructive manipulation;
```

This finally leads us to the main result about unweighted destructive manipulation:

Theorem 6.3. *Let (V, M, p) be an unweighted manipulation instance with m manipulators and $k = \|M\|$ manipulators. Let the defeat strength be monotonic.*

1. *Algorithm 9 computes an unweighted constructive unique winner manipulation's ballot for all manipulators in $\mathcal{O}(n^2)$ time and $\mathcal{O}(n^2)$ space, if one exists.*
2. *An unweighted constructive unique winner manipulation can be computed in $\mathcal{O}(n^2(m + k))$ time and $\mathcal{O}(kn^2)$ space.*
3. *If there is an unweighted constructive unique winner manipulation, then all manipulators can vote the same complete ranking to successfully manipulate.*

Proof.

1. Let $D = D(V)$. Since this is the construction used in the proof of Theorem 6.2, Algorithm 9 computes an unweighted destructive manipulation, if one exists, and otherwise returns an error. It takes $\mathcal{O}(n^2)$ time to find such a required path, $\mathcal{O}(n^2)$ time to compute Q and $\mathcal{O}(n^2)$ to construct the ranking. The space requirement is dominated by the Condorcet pairwise matrix, which takes $\mathcal{O}(n^2)$ space.
2. The Condorcet pairwise matrix can be computed in $\mathcal{O}(mn^2)$ time and the number of manipulators can be counted in $\mathcal{O}(k)$ time. Algorithm 9 takes $\mathcal{O}(n^2)$ time. Finally, the manipulators' voting profile needs to be constructed. If the vote has to be copied for each manipulator in the profile, this takes $\mathcal{O}(kn^2)$ time and space.
3. Algorithm 9 always computes a complete ranking that should be voted by all manipulators. ☐

6.1.2 Complexity

As a result of the previous section, we know that there is a polynomial time algorithm for destructive manipulation. What remains is to find the complexity class of the destructive manipulation problem. The result is stated in the next theorem:

Theorem 6.4. *Let the defeat strength be monotonic.*

The UNWEIGHTED DESTRUCTIVE MANIPULATION *problem is NL-complete for any number of manipulators.*

Proof. First, we show that the problem is in NL. In Theorem 6.2, it was shown that we only need to compare the strength of the strongest paths from and to p. Thus, it suffices to compute these strengths. Algorithm 5 in Lemma 4.19's proof can compute these strengths in logarithmic space on a non-deterministic Turing machine. We only need to exchange the defeat strength matrix and we get an algorithm that can compute the strengths as we need them here.

Next, we have to show that the problem is NL-hard. Let k be the number of manipulators. We reduce STCON to the UNWEIGHTED DESTRUCTIVE MANIPULATION problem: We are given a directed graph $G = (V, E)$ and vertices $s, t \in V$. The reduction is as follows:

- Candidates $C = V \uplus E$.
- A voting profile V_{N} with a margins pairwise matrix $M = M(V_{\mathrm{N}})$ such that

  $$M_{ab} = \begin{cases} 2k+6 & a \in V \setminus \{t\} \wedge (\exists c \in V)[(a,c) = b] \\ 2k+6 & b \in V \setminus \{s\} \wedge (\exists c \in V)[(c,b) = a] \\ 4k+6 & a = t \wedge b \neq s \\ 4k+4 & a = t \wedge b = s \\ -M_{ba} & \text{if } M_{ba} \text{ was defined in previous cases} \\ 0 & \text{otherwise} \end{cases}$$

  We can determine such a voting profile in logarithmic space using McGarvey's trick again. Then, all rankings are complete, so all monotonic defeat strength measures become equivalent to the margins measure.
- k manipulators.
- t should lose.

For the margins measure, Theorem 6.2 simplifies: There is a manipulation if and only if there is a candidate $c \in C \setminus \{t\}$ with $S_M(c,t) + k > S_M(t,c) - k$. Observe that to all candidates c other than s there is a (t,c)-path of strength $4k+6$, and that there is no path to t with a strength greater than $2k+6$. Therefore, the inequality does not hold for all candidates $c \in C \setminus \{s\}$, for $S_M(c,t) + k \leq 2k+6+k = 4k+6-k \leq S_M(t,c) - k$. Therefore, there is a successful manipulation if and only if $S_M(s,t) + k > S_M(t,s) - k$. There is only one link to s with a positive strength, which is the link (t,s) that has strength $4k+4$. Therefore, $S_M(t,s) = 4k+4$, so there is a successful manipulation if and only if $S_M(s,t) > 2k+4$. There is such a strong path if and only if there is an (s,t)-path in G, for there must be an (s,t)-path $P = (v_0, (v_0,v_1), v_1, \ldots, (v_{\ell-1}, v_\ell), v_\ell)$ with strength $s_M(P) > 2k+4$, where $v_i \in V$ and $(v_i, v_{i+1}) \in E$ for all $i \in \{0, \ldots, \ell-1\}$. Hence, there is a successful manipulation if and only if there is an (s,t)-path in G. □

6.2 Weighted Voting

The manipulators may also need to manipulate elections when some voters have votes with a greater weight than others. The problem to manipulate an election with weighted voting is considered in this section.

Obviously, WEIGHTED DESTRUCTIVE MANIPULATION is at least as difficult as UNWEIGHTED DESTRUCTIVE MANIPULATION, for unweighted voting is a special case of weighted voting where all voters have the same weight.

Theorem 6.5. *Let* $I = (V_{\mathrm{N}}, W_{\mathrm{N}}, M, W_{\mathrm{M}}, p)$ *be a weighted manipulation instance with* $k = \|M\|$ *manipulators. Let the defeat strength be monotonic.*

Let M' *be a set of manipulators of size* $\|M'\| = \sum_{i=1}^{k}(W_{\mathrm{M}})_i$*. There is a weighted destructive manipulation for* I *if and only if there is an unweighted destructive manipulation for unweighted manipulation instance* $I' = (E(V_{\mathrm{N}}, W_{\mathrm{N}}), M', p)$.

Proof. The proof is analogous to the proof of Theorem 4.22: It was shown in Theorem 6.3 that all manipulators can vote identically. □

Consequently, the WEIGHTED DESTRUCTIVE MANIPULATION and the UNWEIGHTED DESTRUCTIVE MANIPULATION decision problems are equally hard, at least as long as the weights are not excessively great. If the weights, however, grow too fast, we must reassure that the problem is in NL. Finally, the algorithms to find an unweighted destructive manipulation can straightforwardly be adapted to compute a weighted destructive manipulation, based on the construction used in the proof of Theorem 6.5: We simply assume that there are $\sum_{i=1}^{k}(W_{\mathrm{M}})_i$ manipulators.

The following theorem summarizes these thoughts:

Theorem 6.6. *Let* $I = (V_{\mathrm{N}}, W_{\mathrm{N}}, M, W_{\mathrm{M}}, p)$ *be a weighted manipulation instance with* m *non-manipulators and* $k = \|M\|$ *manipulators.*

1. *The* WEIGHTED DESTRUCTIVE MANIPULATION *problem is* NL*-complete.*
2. *A weighted destructive manipulation can be computed in* $\mathcal{O}(n^2(m+k))$ *time and* $\mathcal{O}(kn^2)$ *space, if one exists.*

Proof.

1. First, the problem must be NL-hard for the special case of unweighted votes is also NL-hard. Next, we can use the same algorithm as the one in Theorem 6.4 to test for the existence of a manipulation. However, this algorithm uses logarithmic space only if we can also represent the sum of the weights of some voters in logarithmic space, which is only the case if the weights are bounded by a polynomial in the size of the input, i.e., by a polynomial in n, m and k.

Therefore, we will no longer represent the sum of the weights in memory, but instead we will calculate each bit of the sum on demand. Algorithm 10 is an algorithm that calculates the kth bit of an iterated addition in logarithmic space. Note especially that carry has a maximum value of n and sum has a maximum value of $2n$ in the algorithm, so both can be represented in logarithmic space.

As an additional tweak, we will also no longer save the currently known lower bound on the path strength directly in memory, but instead will simply save a reference to a link with this strength. Since comparing two values according to the defeat strength is assumed to be a non-deterministic logarithmic space operation, too, the tweaked algorithm is also in NL.

2. The Condorcet pairwise matrix for (V_N, W_N) can be computed in $\mathcal{O}(mn^2)$ time. The total weight of the manipulators' votes can be computed in $\mathcal{O}(k)$ time. A complete ranking which can be voted by all manipulators can be computed in $\mathcal{O}(n^2)$ time using Algorithm 9, using the total weight of the manipulators as the argument for the number of manipulators. Finally, the manipulators' voting profile has to be produced, which takes at most $\mathcal{O}(kn^2)$ time and space. ☐

```
Algorithm 10: Algorithm For Iterated Addition of Non-Negative Integers
Input: Binary numbers $I[1, ..., n]$ with most significant bit $m$
       the $k$th bit to calculate
Output: The $k$th bit and an indicator if there is a non-zero bit of greater
        significance than the $k$th bit
begin
    carry ← 0;
    for $i = 0$ to $k$ do
        sum ← carry;
        if $i \leq m$ then
            sum ← sum + $\sum_{j=1}^{n} I[j]_i$;
        bit ← mod(sum, 2);
        carry ← $\lfloor \frac{sum}{2} \rfloor$;
    return (bit, (carry > 0) ∨ ($k < m$));
```

7 Conclusion

In this thesis, we discussed how difficult it is to manipulate elections under Schulze voting. We concentrated on three special kinds of manipulations that voters could attempt: In Chapter 4 and Chapter 5, we looked into two variants of constructive manipulation; in Chapter 6, we considered a variant of destructive manipulation.

As the main contribution, we developed polynomial time algorithms for constructive manipulation in Chapter 4 and Chapter 5 for coalitions, a problem for which no polynomial time algorithm was known previously except for the special case of only one manipulator in the unweighted setting (cf. [PX12]). Additionally, we also proposed an alternative proof and algorithm for destructive manipulation.

To complement these results, we also studied the complexity of the corresponding decision problems. In Chapter 4, we proved that CONSTRUCTIVE UNIQUE WINNER MANIPULATION, both in the weighted and unweighted case, is essentially P-complete. Only in the special case of UNWEIGHTED CONSTRUCTIVE UNIQUE WINNER MANIPULATION where all voters are required to vote complete rankings and there is one manipulator, the problem was NL-complete.

Similarly, we also showed in Chapter 5 that CONSTRUCTIVE CO-WINNER MANIPULATION is P-complete both in the unweighted and weighted setting. Both results imply that constructive manipulation can probably not be solved efficiently by a parallel algorithm, at least in the worst case.[1]

Finally, we proved that destructive manipulation is an NL-complete problem in the unweighted and weighted setting. This is even true if the weights in the weighted setting are growing extraordinarily fast. As a consequence, there are efficient parallel algorithms that solve these decision problems.

There are still some open questions that could be further pursued:

1. Can the running time of the algorithms that compute constructive manipulations be improved? If we consider the proof of Lemma 4.12, we can see that we can reduce the number of checks for applications of the Restricting Link rule to one check per link: Always only (try to) apply the rule from source candidates that are guaranteed to become stable afterwards. This reduces the time that we spend on the Restricting Link rule to $\mathcal{O}(n^2)$. However, the main obstacle to improve the running time of the algorithm is the Incoming Paths

---

[1] This assumes that the class of problems with efficient parallel algorithms, NC, does not equal the class of problems that can deterministically be solved in polynomial time, P.

rule: As explained, applying the rule can be considered solving a strongest path problem. The weights on some edges of the underlying graph do change with each iteration, which may force us to recompute the strongest paths in each iteration. Furthermore, one can find instances where the number of iterations is linear in the number of candidates. This makes it difficult to spend $o(n^3)$ time on the Incoming Paths rule. For a faster algorithm, the rule will probably have to be replaced or an algorithm that exploits a different property of constructive manipulations of the Schulze method needs to be found.

2. The complete information requirement is a rather strong one. Are there efficient manipulation algorithms for the Schulze method in incomplete information scenarios? Specifically, is there an efficient algorithm that maximizes the probability of manipulation success? This problem would need to be considered for different probability distributions on the voting profiles.

Acknowledgements

I wish to express my gratitude to my supervisor Dr. Sven Kosub, for his support, patience and helpful advice during the research.

I would also like to thank Menatallah el-Assady, Alexander Jäger, Manuel Müller and Leonard Wörteler for their efforts to understand this work and for providing valuable suggestions and corrections.

Bibliography

[Arr50] K. Arrow. "A Difficulty in the Concept of Social Welfare". In: *The Journal of Political Economy* 58.4 (1950), pp. 328–346.

[Bla48] D. Black. "On the rationale of group decision-making". In: *The Journal of Political Economy* 56.1 (1948), pp. 23–34.

[BO91] J. J. Bartholdi and J. B. Orlin. "Single transferable vote resists strategic voting". English. In: *Social Choice and Welfare* 8 (4 1991), pp. 341–354. ISSN: 0176-1714. DOI: `10.1007/BF00183045`.

[BTT89] J. J. Bartholdi, C. A. Tovey, and M. A. Trick. "The computational difficulty of manipulating an election". English. In: *Social Choice and Welfare* 6 (3 1989), pp. 227–241. ISSN: 0176-1714. DOI: `10.1007/BF00295861`.

[CSL07] V. Conitzer, T. Sandholm, and J. Lang. "When are elections with few candidates hard to manipulate?" In: *J. ACM* 54.3 (June 2007). ISSN: 0004-5411. DOI: `10.1145/1236457.1236461`.

[Dij59] E. Dijkstra. "A note on two problems in connexion with graphs". English. In: *Numerische Mathematik* 1 (1 1959), pp. 269–271. ISSN: 0029-599X. DOI: `10.1007/BF01386390`.

[Gib73] A. Gibbard. "Manipulation of voting schemes: a general result". In: *Econometrica: journal of the Econometric Society* (1973), pp. 587–601.

[HH07] E. Hemaspaandra and L. A. Hemaspaandra. "Dichotomy for voting systems". In: *Journal of Computer and System Sciences* 73.1 (2007), pp. 73 –83. ISSN: 0022-0000. DOI: `10.1016/j.jcss.2006.09.002`.

[HLM12] L. A. Hemaspaandra, R. Lavaee, and C. Menton. "Schulze and Ranked-Pairs Voting are Fixed-Parameter Tractable to Bribe, Manipulate, and Control". In: *CoRR* abs/1210.6963 (2012).

[Imm88] N. Immerman. "Nondeterministic space is closed under complementation". In: *SIAM Journal on Computing* 17.5 (1988), pp. 935–938.

[McG53] D. McGarvey. "A theorem on the construction of voting paradoxes". In: *Econometrica: Journal of the Econometric Society* (1953), pp. 608–610.

[Mou80] H. Moulin. "On strategy-proofness and single peakedness". In: *Public Choice* 35.4 (1980), pp. 437–455.

[MS12] C. Menton and P. Singh. “Manipulation and Control Complexity of Schulze Voting”. In: *CoRR* abs/1206.2111v3 (2012).

[OE11] S. Obraztsova and E. Elkind. “On the complexity of voting manipulation under randomized tie-breaking”. In: *Twenty-Second International Joint Conference on Artificial Intelligence.* 2011.

[OEH11] S. Obraztsova, E. Elkind, and N. Hazon. “Ties matter: Complexity of voting manipulation revisited”. In: *The Tenth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2011).* 2011, pp. 71–78.

[Pap94] C. H. Papadimitriou. *Computational Complexity.* Addison-Wesley, 1994.

[PX12] D. Parkes and L. Xia. “A complexity-of-strategic-behavior comparison between Schulze’s rule and ranked pairs”. In: *Proceedings of the 26th AAAI Conference on Artificial Intelligence (AAAI). AAAI Press. Forthcoming.* 2012.

[Sat75] M. Satterthwaite. “Strategy-proofness and Arrow’s conditions: Existence and correspondence theorems for voting procedures and social welfare functions”. In: *Journal of economic theory* 10.2 (1975), pp. 187–217.

[Sch11] M. Schulze. “A new monotonic, clone-independent, reversal symmetric, and condorcet-consistent single-winner election method”. In: *Social Choice and Welfare* 36.2 (2011), pp. 267–303.

[Sze88] R. Szelepcsényi. “The method of forced enumeration for nondeterministic automata”. English. In: *Acta Informatica* 26 (3 1988), pp. 279–284. ISSN: 0001-5903. DOI: `10.1007/BF00299636`.

[Xia+09] L. Xia et al. “Complexity of unweighted coalitional manipulation under some common voting rules”. In: *In Proc. of IJCAI-09.* 2009.